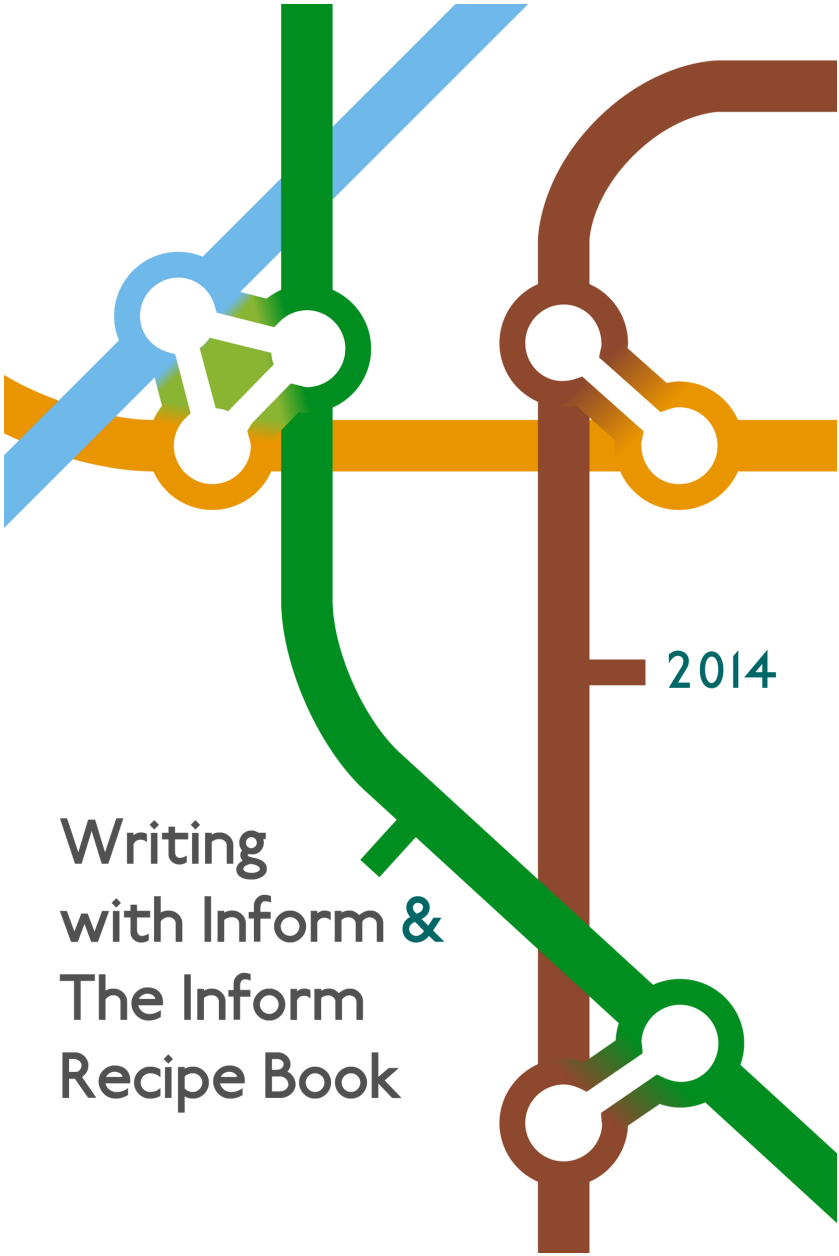


2014

Writing with Inform & The Inform Recipe Book

Part I. Writing with Inform

Chapter 1: Welcome to Inform
Chapter 2: The Source Text
Chapter 3: Things
Chapter 4: Kinds
Chapter 5: Text
Chapter 6: Descriptions
Chapter 7: Basic Actions
Chapter 8: Change
Chapter 9: Time
Chapter 10: Scenes
Chapter 11: Phrases
Chapter 12: Advanced Actions
Chapter 13: Relations
Chapter 14: Adaptive Text and Responses
Chapter 15: Numbers and Equations
Chapter 16: Tables
Chapter 17: Understanding
Chapter 18: Activities
Chapter 19: Rulebooks
Chapter 20: Advanced Text
Chapter 21: Lists
Chapter 22: Advanced Phrases
Chapter 23: Figures, Sounds and Files
Chapter 24: Testing and Debugging
Chapter 25: Releasing
Chapter 26: Publishing
Chapter 27: Extensions

Start reading here: §1.1. Preface
Part II. The Inform Recipe Book
Indexes of the examples and definitions

Chapter 1: Welcome to Inform

1.4. The Go! button; *§1.5. The Replay button*; *§1.6. The Index and Results panels*; *§1.7. The Skein*; *§1.*

Contents of *Writing with Inform*
Chapter 2: The Source Text
Indexes of the examples

§1.1. Preface

Welcome to Inform, a design system for interactive fiction based on natural language.

Interactive fiction is a literary form which involves programming a computer so that it

presents a reader with a text which can be explored. Inform aims to make the burden of learning to program such texts as light as possible. It is a tool for writers intrigued by computing, and computer programmers intrigued by writing. Perhaps these are not so very different pursuits, in their rewards and pleasures.

> The sheer joy of making things... the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles... the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. (Frederick P. Brooks, "The Mythical Man-Month", 1972)

Writing with Inform is one of two interlinked books included with Inform: a concise but complete guide to the system. The other book is **The Inform Recipe Book**, a comprehensive collection of examples, showing its practical use. If you are reading this within the Inform application, you will see that the Writing with Inform pages are on "white paper", while the Recipe Book is on "yellow paper".

These notes are arranged so that the reader can, in principle, write whole works of fiction as early as the end of Chapter 3. Each subsequent chapter then extends the range of techniques available to make livelier and more intriguing situations.

This new release of Inform ("Inform 7", the seventh major version since 1993) is a radical departure from most previous approaches to interactive fiction. In particular, it is very different from Inform 6, which newcomers will not need to know anything about. Inform 6 sits inside Inform 7, and is part of the inner workings, but is not visible from the outside. For information about Inform 6, see **www.inform-fiction.org**.

> Programming is best regarded as the process of creating works of literature, which are meant to be read... so we ought to address them to people, not to machines. (Donald Knuth, "Literate Programming", 1981)

See **Acknowledgements** *for a chance to try out the cross-referencing links in Writing with Inform - click on the red asterisk or the name of the destination to go there*

Start of Chapter 1: Welcome to Inform
Onward to §1.2. Acknowledgements
Example 1: **About the**

examples An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

---

§1.2. Acknowledgements

Inform 7 is dedicated to Emily Short and Andrew Plotkin, whose shrewd and sceptical suggestions made a contribution which can hardly be overstated. A long email correspondence with Andrew entirely subverted my original thoughts about natural-language IF, as he convinced me that the "new model" of rule-based IF was a truer foundation; while Emily's wry, witty analysis and how-about-this? cheered me at low moments, besides providing the impetus and often the specifics for a lot of the best ideas.

From the outset, I have thought of Inform 7 as no longer being a command-line compiler, but a compiler in combination with a humanising user interface. All credit for the reference implementation under Mac OS X belongs to Andrew Hunter. How simple the metaphor of an interactive book with facing pages may seem, but the coding was an enormous challenge. In 2014 Toby Nelson, the author's brother, put months of time into the project by rewriting and modernising the Mac OS X application: sandboxing it for the Mac App Store, giving it a more contemporary design, and much more.

Though David Kinder's Windows application does indeed visually follow the OS X original, the two programs were coded independently, and the programming task taken up by David was formidable indeed. Philip Chimento's Gnome-based user interface for Linux became officially part of the project in November 2007, when the first easy-to-install packages for Ubuntu and Fedora were offered. Philip's efforts were particularly generous since the early stages of Inform-for-Linux were so tentative: for many months, we weren't sure how to go about the project, and during that time Philip quietly wrote us a solution. Adam Thornton continues to support Inform at the command line on Unix-like systems.

Inform in its widest sense incorporates work by so many people that it's simply impossible to thank all of them, but Erik Temple, Dannii Willis, Ron Newcomb, Eric Eve and Juhana Leinonen all deserve special mention. More than 250 users have filed patient and careful bug reports, keeping us on the straight and narrow. They're contributors, too.

It's perhaps surprising that the Inform application gained its first online component only in 2014, but the Public Library of Extensions, and its discussion forum, make a brave step into the modern age. Mark Musante, our Extensions librarian, Dannii Willis and Justin de Vesine have been a great help in setting this up.

The original development of Inform 7 was a long haul, and I would particularly like to thank Sonja Kesserich, David Cornelson and other volunteers for their early testing of a then-fragile system. The final months before the Public Beta release of Inform 7 were made more enjoyable, as well as more productive, by fruitful discussions leading to a cross-platform standard for bibliographic data and cover art. L. Ross Raszewski, who wrote frighteningly efficient reference software in frighteningly little time; the librarians of the IF-Archive, Andrew Plotkin, David Kinder and Paul Mazaitis; and my fellow authors of IF design systems - Mike Roberts (of the Text Adventure Development System); Kent Tessman (of Hugo); and Campbell Wild (of ADRIFT).

This EPUB edition of the documentation was greatly assisted by excellent advice published by Liza Daly, an old friend of Inform's who also helped construct our website.

Start of Chapter 1: Welcome to Inform

Back to §1.1. Preface

Onward to §1.3. The facing pages

§1.3. The facing pages

On most computers, Inform runs in a single main window which is an opened book showing two facing pages. As we shall see it behaves as if these pages are in dialogue with each other: for the most part we write on the left hand page and see responses appear on the right. But all is controllable. The margin between the two pages can be dragged back and forth like the slide on a trombone: each page can be made smaller that the other may grow larger. Moreover, each page can display one of a number of displays relevant to the current project, called "panels", one of them being the Documentation panel which displays a screen-readable copy of this manual. The vertical strip of choices at the right hand margin of each page allows you to choose between panels. (The same panel can be showing on both pages at the same time, if that's useful.)

At the start the only panels available are a blank space in which to write the first lines of a new interactive fiction - the Source panel - and this one, the Documentation. Clicking on the other choices will do nothing.

The exception is the Settings panel, which contains some preference settings for the individual project - not the whole application. This is always available, but it controls settings which can be left alone almost all of the time.

Start of Chapter 1: Welcome to Inform

§1.4. The Go! button

Clicking the Go button translates the text in the Source panel into a computer program which enacts the interactive fiction, and automatically sets it going (in the Story panel, which opens as needed).

If the Source is empty of text, Inform will be unable to create anything: it needs at least one name of a location where the drama can unfold. For reasons of tradition, such locations are normally called "rooms", though people have used them to represent anything from grassy fields to states of mind and other metaphorical places.

```
"Midsummer Day"

The Gazebo is a room.
```

Clicking Go with this text in the Source panel will result in a short delay, after which the Story panel will appear, from which we can explore this newly created world: an interactive fiction called "Midsummer Day". It will not be very exciting, since Inform has only five words to go on, but we can add more detail to the source at any point and then click Go again to try out the changes. (Note that there is no need to "quit" these explorations in the Story panel. When Go is clicked, any story already in progress is discarded in favour of the new version.)

The keyboard shortcut Command-R (on Mac OS X), F5 (on Windows), or Ctrl-R (on Linux GNOME) has the same effect as clicking Go.

Start of Chapter 1: Welcome to Inform
Back to §1.3. The facing pages
Onward to §1.5. The Replay button

§1.5. The Replay button

Replay works identically to Go, except that it does something further: once the story is created, it automatically plays through the same commands as were typed into the previous version. For instance: suppose we click Go to bring Midsummer Day into being, and find ourselves playing the story. We type "look" and find that there is not much to see. Going back to the source, we add

```
"A white canvas parasol raised up on stakes driven into the grass."
```

so that the source now reads

```
"Midsummer Day"

The Gazebo is a room. "A white canvas parasol raised up on stakes
driven into the grass."
```

Instead of clicking Go, we click Replay, and can sit back and watch what has changed. In this example, it only saves us the trouble of typing "look", but once stories become long and elaborate, Replay is invaluable: and especially when we notice in play that something very minor is wrong - a spelling error, say - and want to fix it immediately, without fuss.

Start of Chapter 1: Welcome to Inform
Back to §1.4. The Go! button
Onward to §1.6. The Index and Results panels

§1.6. The Index and Results panels

If, when Go! is clicked, the text in the Source panel is not fully understood, then Inform will generate a report of the problems it found, which will open in the "Errors" panel. (Other information is also available in "Errors", but most of it is used for debugging Inform, and can be ignored.)

On the other hand, if the text was fully understood then another new panel will become available: the "Index". This is a cross-referenced index of the source, or rather, of the interactive fiction which has been generated. The Index is only an optional convenience, but becomes more and more helpful as the fiction grows larger.

Its exact format does not matter for now.

The icon always denotes a reference to a particular line in the Source text, that is, to something written in the source: clicking it opens the Source panel and jumps to that position.

The icon indicates that more detailed information can be read further down the text in the same panel: clicking it jumps down to this more detailed report.

Lastly, the icon hints that there is a relevant page of this manual: clicking this opens the Documentation panel and switches to it.

Start of Chapter 1: Welcome to Inform

Back to §1.5. The Replay button

Onward to §1.7. The Skein

---

§1.7. The Skein

The Replay button demonstrates that Inform must be quietly remembering the commands typed into the last run through the story. In fact it remembers, and automatically organises, *every* previous run.

Inform's approach to testing interactive fiction is to treat it as being like the analysis of other turn-based games, such as chess. It would be prohibitively difficult to work out every possible combination of moves: instead, we analyse those which go somewhere, and look for significant choices. Every Queen's Gambit begins with the same first three moves (1. d4, d5; 2. c4), but then there is a choice, as the next move decides whether we have a Queen's Gambit Accepted (dxc4) or Declined (e6). Books about chess often contain great tables of such openings, which run together for a while but eventually diverge. To learn chess, one must explore all of these variations.

Inform's Skein panel is just such a table, built automatically. If we think of the list of typed commands as a thread, then the skein is (as the name suggests) braided together from all these threads. In the display, time begins at the top, with the **start** knot, and the threads of different play-throughs hang downwards from it.

Double-clicking on a command translates the source afresh and replays the story from **start** down to that command, and then stops. We are then free to continue play by typing commands into the Story panel, of course, and these commands will automatically be recorded in the Skein as a new variation of play, diverging from the previous threads.

Start of Chapter 1: Welcome to

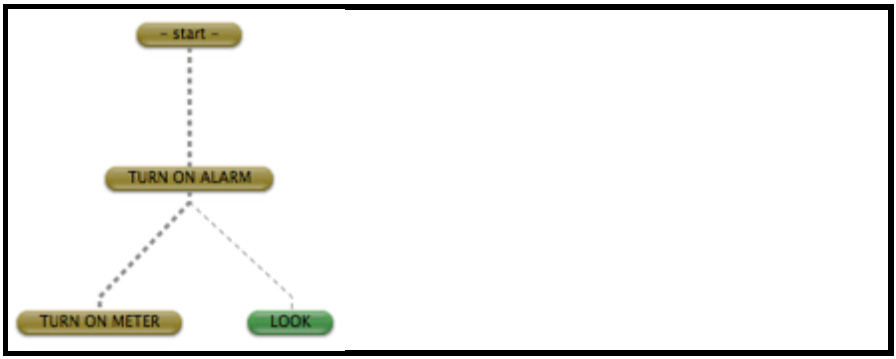
§1.8. A short Skein tutorial

In the following example, we will see how the Skein is woven as different commands are tried. As it happens, the story being played is the example "Witnessed", from Chapter 11, but the details do not matter. When the project has never been played at all, if we switch to the Skein panel (or open it opposite the Story panel) we will only find this:

Suppose we click Go for the first time and type two commands in: TURN ON ALARM and then LOOK. Now the Skein shows:

Only one line of play is known to Inform, and it runs downwards in a thread from the special "- start -" knot, which represents the situation before any command has been tried. The useful thing about having past histories recorded like this is that we can revisit them. Suppose we want to go back to the situation after typing only TURN ON ALARM. We could click Go again and type that first command in once more, but now we have an easier method: we simply double-click on the TURN ON ALARM knot. The story restarts by itself, and commands are automatically keyed in to regain the position of play represented by the knot we clicked on - in this example that only keys a single command in, but it might have been hundreds. The Skein now looks like this:

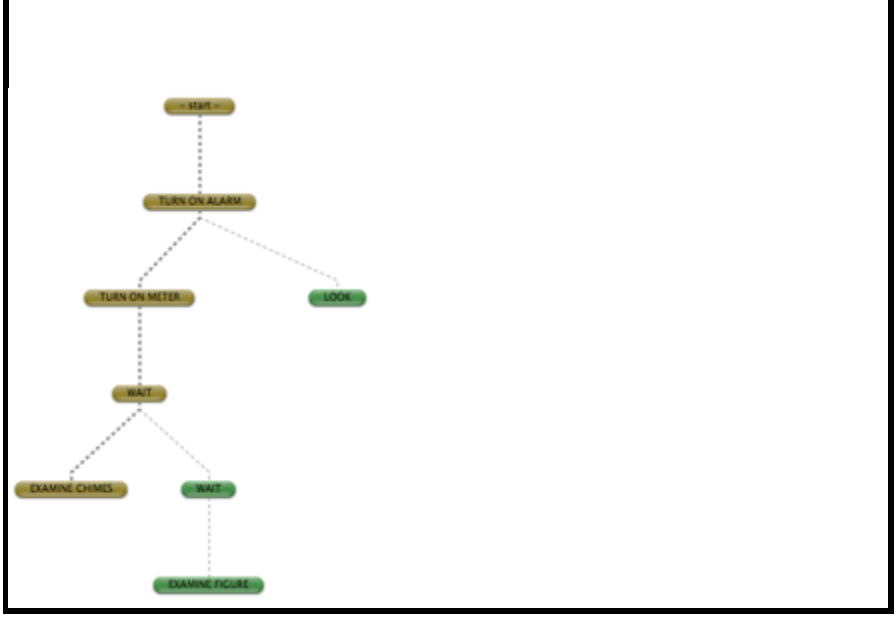
All knots are displayed either as yellow or green. Yellow knot are the ones in the history of the story currently playing. The LOOK knot is green because it hasn't happened in the current story yet - and in fact, it won't happen in the current story, because instead we play TURN ON METER. Now the Skein changes again:



Inform now knows about two ways to play the current project: one consisting of TURN ON ALARM and then TURN ON METER, the other of TURN ON ALARM and then LOOK. Since these only differ after the first turn, Inform displays them as a thread which divides into two after the first turn. Again, LOOK remains green because it hasn't been played in the current story.

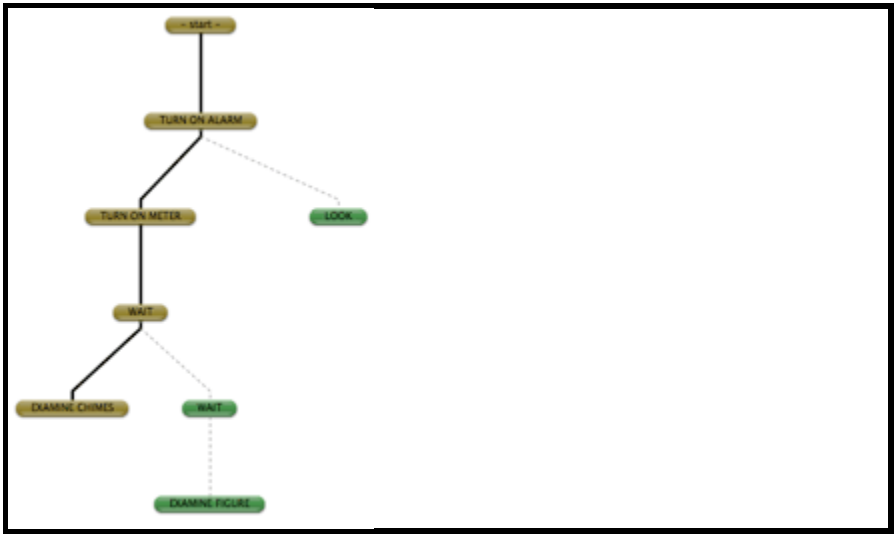
Note also that one of the two possible threads here is drawn more thickly (here it is shown with thick dashes rather than thin). Only one thread is ever drawn thickly -- the one currently being shown in the Transcript panel, which we will come to later on. (That often corresponds to the current line of play, as now, because the Transcript follows what we do unless we choose otherwise.)

After a little more exploration, we reach the following:



At this point we decide that we want to preserve the thread leading to EXAMINE CHIMES - perhaps it's a sequence we are going to want to test often. The Skein can be edited very easily: right-clicking on a knot brings up a contextual menu.

We choose Lock This Thread from the contextual menu, and this makes the thread through to here "locked". That means the knots can't be deleted (unless we unlock them again) - either by our own mistake, or by Inform trimming back no-longer-needed threads of the Skein to keep it manageable in size.

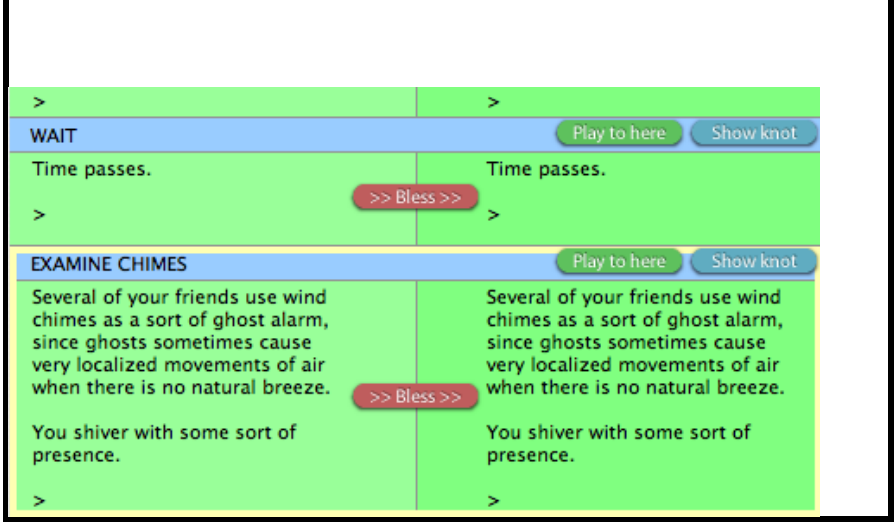


Note that this locked history is now drawn as a solid thread, whereas all the others are unlocked and drawn as dashes.

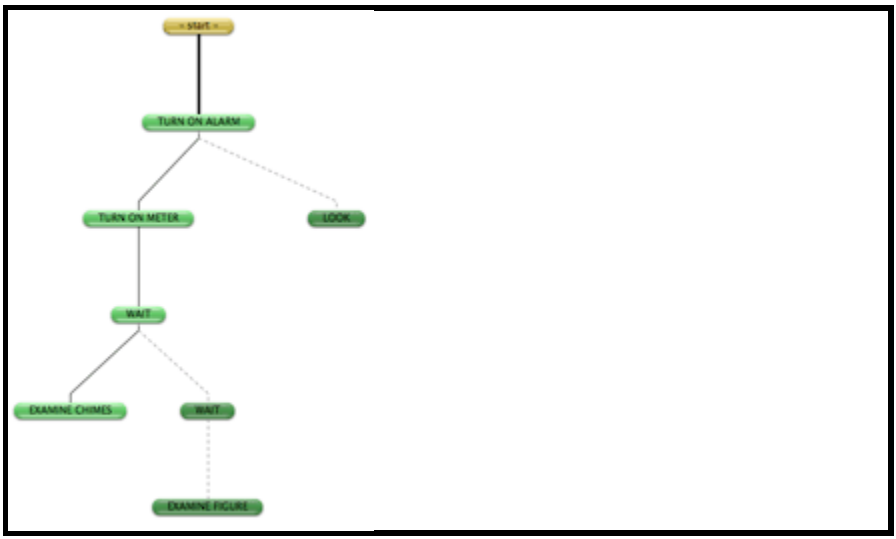
Now we have a securely remembered piece of standard play: it means we can try out the sequence TURN ON ALARM / TURN ON METER / WAIT / EXAMINE CHIMES any time we want to with a double-click on the final knot. This is convenient for testing - but so far it only runs the test: to see whether the test came out well or badly, we have to look through what happened, perhaps by scrolling back in the Story panel to look at the text. And that means that we need to remember what the text should have been like.

In fact, though, Inform can remember for us, using the Transcript panel. This is closely joined to the Skein panel, and it's often convenient to flip between the two. Turning to the Transcript now, we find a two-column view of the story currently being played. The left-hand column shows the text which has been displayed on each turn so far; the right-hand column is empty. The bottom of the Transcript looks like so:

The empty right-hand column displays the "blessed" transcript - one which the author has approved as being correct. This can be done for each individual knot, using the Bless button joining the columns, but in this case we will bless the whole transcript of this story, using the Bless All button. Now there's text in both columns, and of course the two columns match. (Note that the blessed transcript is in a brighter colour.)

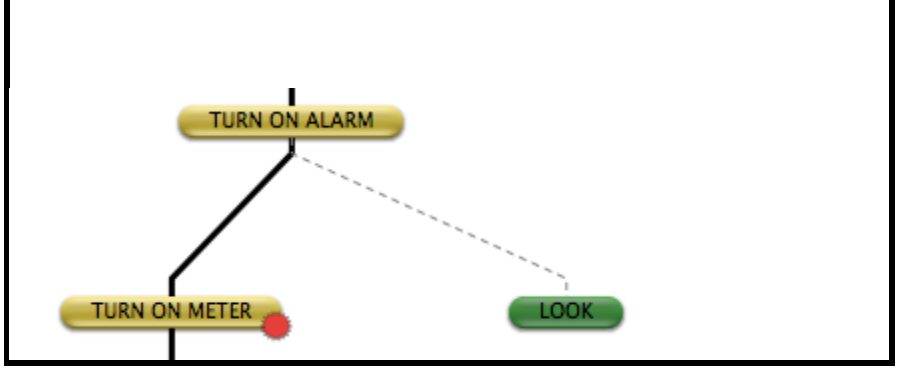


Back in the Skein, we find that the knots which have transcripts have lit up, and are brighter than the others. If we Go, to start a new story, and then look at the Skein:

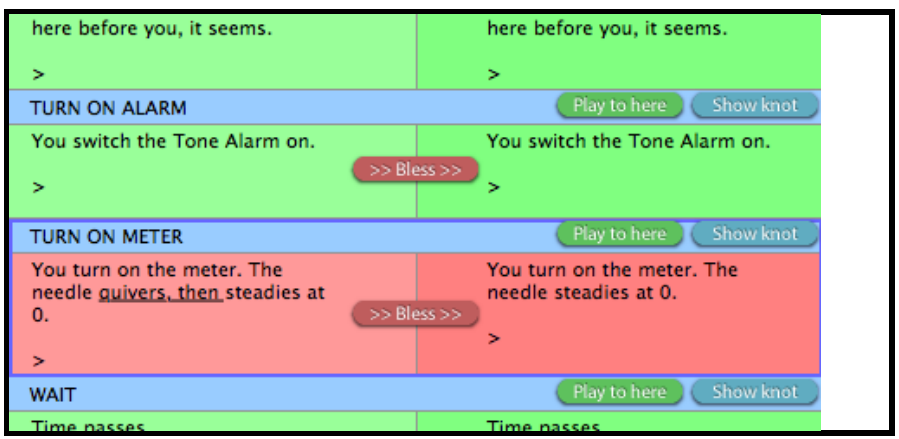


we see that the knots for which we have blessed a transcript are in a brighter green (or a brighter yellow, if they're in the current story being played).

Now suppose we change the source text for the project, so that we make it behave differently. The details don't matter, but suppose we do something which changes the result of the TURN ON METER command, and then run the test again. Now we find:



The red warning badge on the TURN ON METER knot alerts us that the last time this knot was tried (just now, as it happens), the resulting text didn't agree with its blessed transcript. (Red badges can only be seen on bright-coloured knots which have transcripts - for other knots, there's nothing to compare with.) On the other hand, the rest of the yellow current line of play worked out exactly the same as we expected - so no badges. Clicking on the red badge takes us into the Transcript panel at the right place, where the corresponding turn's transcript has also turned red:

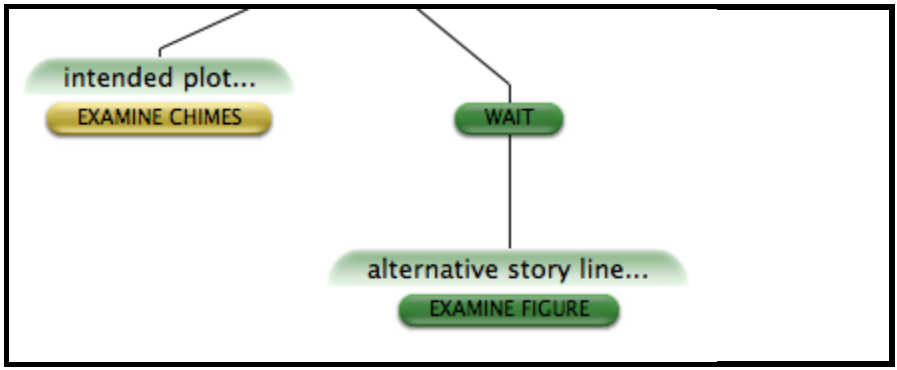


Again, what actually happened is on the left; what should have happened is on the right. The change is shown with underlining - we added the text "quivers, then". If we approve this change, by clicking on the Bless button for the red turn, the amended text will become the correct text to compare against in future runs, and the turn will become green to show that once again all is well. (We can also edit the blessed transcript directly, by double-clicking in the text and typing.) Clicking on the Show knot button takes us back in the skein, at the right place: where we will see that the red warning badge has disappeared.

Some writers of IF like to work backwards from a transcript of the story they want to produce, and for them, the Skein and Transcript combination will be helpful as a running picture of what works so far. Other authors may not use the Skein/Transcript feature at all until right at the end of a project, in testing before publication, when it becomes very important to be able to make small changes in one area without upsetting everything else. Either way, the Skein and Transcript together make a very

powerful testing aid.

This tutorial has shown only a short line of play, to keep the pictures small, but for a large project the Skein might run to thousands of knots. It then becomes important to be able quickly to find key knots corresponding to plot developments. To help with that, we can annotate certain knots with any label we choose (by selecting Add Label from the contextual menu):



And this is where the "Labels" gadget at the top of the Skein comes into its own:

since it offers a menu of all the labels in the Skein, and if selected will jump to the one chosen.

The Skein has other abilities too, best explored by experimenting. For instance, we can edit the commands by selecting Edit Knot from the contextual menu, and we can add new knots in the middle of existing lines using the Insert Knot item on that menu. The Play All Blessed option (on the Game menu) is especially powerful: it tests each possible blessed history in turn, trying all of them, and can therefore test very complicated multiple endings and the like in a single click.

Start of Chapter 1: Welcome to Inform

Back to §1.7. The Skein

Onward to §1.9. Summary of the Skein and Transcript

§1.9. Summary of the Skein and Transcript

The Skein records the history of different plays through the current project, and the Transcript records the text of each response, comparing it with a "blessed" or correct version if one is available.

In the Skein each typed command is a "knot". The threads hanging down from the top "- start -" knot are possible histories. Double-click on a knot to play through to there.

Yellow knots are commands played so far in the current story: green knots are possible lines not taken, or not taken yet.

A solid thread is "locked" and protected from deletion (by accident or when Inform trims away loose ends): a dashed thread has no such protection.

A bright knot has a blessed transcript: a darker knot is one which has no blessed transcript. When a bright knot shows a red badge, this means that when last tested its command produced a textual reply which wasn't the same as the blessed transcript. Clicking on the badge shows exactly how.

The thicker thread in the Skein shows the history currently being displayed in the Transcript panel.

Start of Chapter 1: Welcome to Inform

Back to §1.8. A short Skein tutorial

Onward to Chapter 2: The Source Text: §2.1. Creating the world

---

Examples from Chapter 1: Welcome to Inform

Start of this chapter

Chapter 2: The Source Text

Indexes of the examples

Example **About the examples** *RB*

An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

> This is the first of about 400 numbered examples. In a few cases, such

as this one, they provide a little background information, but almost all demonstrate Inform source text. The techniques demonstrated tend to be included either because they are frequently asked for, or because they show how to achieve some interesting effect.

The same examples are included in **both** of the books of documentation, but in a different order: in *Writing with Inform*, they appear near the techniques used to make them work; in *The Inform Recipe Book*, they are grouped by the effects they provide. For instance, an example called "Do Pass Go", about the throwing of a pair of dice, appears in the "Randomness" section of *Writing with Inform* and also in the "Dice and Playing Cards" section of *The Inform Recipe Book*. Clicking the italicised WI and RB buttons at the right-hand side of an example's heading switches between its position in each book.

Many computing books quote excerpts from programs, but readers have grown wary of them: they are tiresome to type in, and may only be fragments, or may not ever have been tested. The authors of Inform have tried to avoid this. All but two dozen examples contain entire source texts. A single click on the paste icon (always placed just left of the double-quoted title) will write the complete source text into the Source panel. All that is then required is to click the Go button, and the example should translate into a working game.

In most cases, typing the single command TEST ME will play through a few moves to show off the effect being demonstrated. (You may find it convenient to create a "scratch" project file for temporary trials like this, clearing all its text and starting again with each new test.)

As part of the testing process which verifies a new build of Inform, each example in turn is extracted from this documentation, translated, played through, and the resulting transcript mechanically checked. So the examples may even work as claimed. But the flesh is weak, and there are bound to be glitches. We would welcome reports, so that future editions can be corrected.

Each example is loosely graded by difficulty: if they were exercises in a textbook, the asterisks would indicate how many marks each question scores. As a general rule:

★ - A simple example, fairly easily guessed.
★★ - A complicated or surprising example.
★★★ - An example needing detailed knowledge of many aspects of the system.
★★★★ - A complete scenario, containing material not necessarily relevant to the topic being demonstrated.

In general, the main text of *Writing with Inform* tries never to assume knowledge of material which has not yet appeared, but the trickier examples almost always need to break this rule.

Chapter 2: The Source Text

The TEST command; *§2.9. Material not for release*; *§2.10. Installing extensions*; *§2.11. Including extens

Contents of *Writing with Inform*
Chapter 1: Welcome to Inform
Chapter 3: Things
Indexes of the examples

§2.1. Creating the world

Designing an interactive fiction can be divided into two related activities. One is the creation of the world as it appears at the start of play: where and what everything is. The other is to specify the rules of play, which shape how the player interacts with that initially created world. A new Inform project is void and without form, so to speak, with nothing created: but it starts with hundreds of standard rules already in place.

The same division between creating things, and laying down rules, is visible in Inform source text. The creation of the world is done by making unconditional factual statements about it. For example,

```
The wood-slatted crate is in the Gazebo. The crate is a container.
```

Inform calls sentences like these "assertions". The verb is always written in the present tense (thus the crate "is", not "will be"). Further examples are:

```
Mr Jones wears a top hat. The crate contains a croquet mallet.
```

The words "is", "wears" and "contains" are forms of three of the basic verbs built in to Inform. There are only a few built-in assertion verbs, of which the most important are *to be*, *to have*, *to carry*, *to wear*, *to contain* and *to support*. (As we shall see, further assertion verbs can be created if needed.)

The world described by these assertions is the starting condition of the story: what happens when play begins is another matter. If somebody picks up the crate and walks off with it, then it will no longer be in the Gazebo. Mr Jones may remove his hat.

§2.2. Making rules

The other kind of sentence tells Inform what should happen in certain circumstances, and reads like an instruction issued to someone:

```
Instead of taking the crate, say "It's far too heavy to lift."
```

This is a "rule", and it changes the crate's behaviour. The player who tries typing "take crate", "pick up the crate" or similar will be met only with the unhelpful reply "It's far too heavy to lift." The many different kinds of thing which the player can do are called "actions", and are always written as participles: "taking ...", for instance, or "putting ... on ...".

Inform is built on a mass of several hundred rules, some quite complex, and it could even be said that Inform *is* that mass of rules. We never see the complexity behind the scenes because the whole aim is to provide a basic, penny-plain, vanilla flavoured sort of realism. It would be surprising if one could put the crate inside itself, so a rule exists to forbid this. It would be surprising if one could drop something which was already on the ground, and so on. These basic rules of realism are the ones which every new Inform project starts with.

A rule always starts with a situation which it applies to, and then follows with one or more things to do. Here's an example where the situation is "Before taking the crate" - the player is just starting to try to pick the box up - and there's a three-step process to follow, but steps 2 and 3 happen only if step 1 comes out in a particular way:

```
Before taking the crate:
    if the player is wearing the hat:
        now the hat is in the crate;
        say "As you stoop down, your hat falls into the crate."
```

The steps to follow here are called "phrases". Inform knows about 400 built-in phrases, but most of them are needed only occasionally. These three are used over and over again:

if tells Inform to do something only if some "condition" holds, here "the player is wearing the hat";
now tells Inform to change the situation, here so that the hat moves to the crate; and
say tells Inform to say something, that is, to write some text for the player to read.

Every one of the built-in phrases has a definition somewhere in this book. The full definition of "say" will come later, but in the simple form above it writes out the given text for the player to read. (Normally this text is simply shown on screen, not spoken aloud, unless software adapted for partially sighted people is being used.) Phrase definitions are all linked to in the Phrases page of a project's Index.

Start of Chapter 2: The Source Text

Back to §2.1. Creating the world

Onward to §2.3. Punctuation

§2.3. Punctuation

An example rule from the previous section demonstrates one of Inform's conventions about punctuation, and is worth pausing to look at again.

```
Instead of taking the crate, say "It's far too heavy to lift."
```

In English grammar, it's usual to regard a full stop as closing its sentence even when it occurs inside quotation marks, provided there is no indication to the contrary, and this is also the rule used by Inform. Thus:

```
The description is "Shiny." It is valuable.
```

is read as equivalent to

```
The description is "Shiny.". It is valuable.
```

Sentence breaks like this occur only when the final character of the quoted text is a

full stop, question mark or exclamation mark (or one of these three followed by a close bracket) and the next word begins, in the source code, with a capital letter. A paragraph break also divides sentences, behaving as if it were a full stop.

Material in square brackets [like so] is "comment", in computing jargon: it is considered as being an aside, a private note by the author, and not read in by Inform. This allows us to make notes to ourselves like so:

```
The China Shop is a room. [Remember to work out what happens if
the bull gets in here!]
```

Inform is all about text, so pieces of text are often quoted in Inform source. This example is typical:

```
The description is "Shiny." It is valuable.
```

Quotations always use double-quotation marks, which aren't part of the text. So the description here is just the five letters and full stop in between the marks:

```
Shiny.
```

That seems straightforward, but there are three conventions to watch out for.

1. Square brackets [and] inside quoted text don't literally mean [and]. They're used to describe what Inform should say, but in a non-literal way. For example,

```
"Your watch reads [time of day]."
```

might produce

```
Your watch reads 9:02 AM.
```

These are called "text substitutions". They're highly flexible, and they can take many different forms.

2. Single quotation marks at the edges of words are printed as double. So:

```
"Simon says, 'It's far too heavy to lift.'"
```

produces

```
Simon says, "It's far too heavy to lift."
```

3. Texts which end with sentence-ending punctuation - full stop, question mark, exclamation mark - are printed with a line break after them. So:

```
say "i don't know how this ends";
say "I know just how this ends!";
```

would come out quite differently - this doesn't affect the appearance of the text, but only the position where the next text will appear. Something to be careful about is that this only applies when the punctuation occurs at the end of a "say", as in these examples. (It doesn't apply when a varying textual value is printed, using some text substitution, because then the pattern of where line breaks occur would be unpredictable - sometimes the value might end in a punctuation mark, sometimes not.)

These three punctuation rules for texts feel very natural with practice, and Inform users sometimes don't realise the third rule is even there, because it just seems the right thing to happen. But occasionally the rules get in the way of what we want to do. (For instance, how do we get a literal [or]? What if we want a single quote mark where Inform thinks we want a double, or vice versa?) So we'll come back to these rules in more detail in the chapter on Text.

Inform also reads other punctuation marks. Colon ":" and semicolon ";" turned up in the previous section, in the writing of rules. It also has the more exotic "|" (not a capital I, a vertical stroke) for paragraph breaks outside of quoted text, but people hardly ever need this.

As these examples begin to show, Inform source imitates the conventions of printed books and newspapers whenever there is a question of how to write something not easily fitting into words. The first example of this is how Inform handles headings, but to see why these are so useful we first look at Problems.

See **How Inform reads quoted text** *for a fuller exploration of the punctuation rules for text*

Start of Chapter 2: The Source Text

Back to §2.2. Making rules

§2.4. Problems

The language used in the source reads as if it were English aimed at a human reader (and this is intentional: the designer, after all, is a human reader and needs to be able to understand his or her own source), but in reality Inform can only understand a very modest range of sentences and will complain if its limits are passed. Subtler problems arise if the source contains contradictions. For instance, the following "Problem" might be produced:

> **Problem.** You wrote 'A starting pistol is in the cup' , but in another sentence 'A Panama hat is on the cup' : the trophy cup cannot both contain things and support things, which is what you're implying here. If you need both, the easiest way is to make it either a supporter with a container attached or vice versa. For instance: 'A desk is here. On the desk is a newspaper. An openable container called the drawer is part of the desk. In the drawer is a stapler.'

This is a rather discursive error message, and if a similar problem were to occur in the same run through, it would be curtailed to:

> **Problem.** You wrote 'A firing pistol is in the box' , but in another sentence 'A fedora hat is on the box' : again, the croquet box cannot both contain things and support things.

Start of Chapter 2: The Source Text

Back to §2.3. Punctuation

Onward to §2.5. Headings

§2.5. Headings

Once the source grows beyond 1000 words or so, it can all too easily become disorganised, and by the time it reaches the size of a novella it can be difficult to find things. (Though nearly all editors provide a Find function, and the keyboard short-cuts Command-F (Mac OS X) or Ctrl-F (Windows) do the trick in the Inform application.)

Inform provides for us to organise the source code in just the way that a printed book would be organised: with headings and subheadings. Firstly, we can put the title at the top. If the first paragraph consists only of a single quoted piece of text, then that's the title; and an author can also be given, as follows:

```
"Spellbreaker" by Dave Lebling
```

We will later see that more bibliographic information can also be placed here, in the same way that the imprint page of a novel comes before the text gets going. The author's name can normally be given without quotation marks, so long as it contains no punctuation. For instance:

```
"Three Men in a Boat" by "Jerome K. Jerome"
```

needs quotes as otherwise the full stop after the K will be mistaken for the end of a sentence.

A sentence which is the only one in its paragraph and which begins with any of the words "volume", "book", "part", "chapter" or "section" is considered to be a heading or a sub-heading. It must not contain a typed line break, and in order to stand alone in its paragraph there should be a skipped line both before and after it. For instance:

```
Section 2 - Flamsteed's Balloon
```

Headings can be written in any format, provided they start with one of the five indicator words, and they are hierarchical: a "Part ..." heading is considered more significant than a "Chapter ..." heading but not so significant as a "Book ..." heading, and so on. (We do not need to use all five kinds of heading.)

Start of Chapter 2: The Source Text
Back to §2.4. Problems
Onward to §2.6. Why using headings is a good idea

§2.6. Why using headings is a good idea

Reports of problems, as we have seen, often quote back the source to justify

themselves. Rather than quoting line numbers ("Midsummer Day, line 2017" or something similar) Inform uses the icon. The down side of this is that a glance at the list of problems might give little hint of whereabouts in the source the difficulties lie. Inform therefore makes use of headings to give a general indication:

> In Part the First, Chapter 1 - Attic Area:
>
> **Problem.** You wrote 'South of the Attic is the Winery', but in another sentence 'South of the Attic is the Old Furniture': this looks like a contradiction, which might be because I have misunderstood what was meant to be the subject of one or both of those sentences.
>
> In Chapter 2 - Deeper In:
>
> **Problem.** You wrote 'The Disused Observatory is south of the Dark Room', but in another sentence 'South of the Dark Room is the Cupboard': again, this looks like a contradiction.

Secondly, headings are used in the Contents page of the Index, and they allow rapid navigation through the source, by jumping to any heading or subheading with a single click.

Finally, headings are used when working out what a name refers to. Suppose the source contains both a "four-poster bed" and also a "camp bed", and we write something like "The pillow is on the bed." Inform decides which bed is meant by giving priority to whichever is defined in the current section (so far), or failing that the current chapter, or current part, or current book, or finally the current volume. This allows us to write, for instance,

> The four-poster bed is in the Boudoir. The pillow is on the bed.

and not have the pillow mysteriously turn up on the camp bed, which hasn't been mentioned since way back in Chapter 2.

Start of Chapter 2: The Source Text

§2.7. The SHOWME command

Problem messages are generated when the source text does not make sense to Inform. Even if it does make sense, though, there is no guarantee that it does what the author intends, and the only way to find out is to test the result by playing through it (or asking others to). For the most part one plays as if one were the eventual reader of the work, but sometimes it is highly convenient to have the god-like powers which are an author's prerogative. These are provided by the testing commands, which are present at every stage until the final release version (generated by the Release button). They will be introduced in this manual as they become relevant: here is the first.

The testing command SHOWME prints out a brief summary about a room or thing, and any contents or parts it may have. Typing SHOWME on its own shows the current room, but any item or room in the story, however distant, can be named instead. For instance:

```
>showme
Boudoir - room
   four-poster bed - supporter
   yourself - person
   pillow
```

```
>showme diamonds
diamonds - thing
location: in the strongbox on the dresser in the Drawing Room
unlit; inedible; opaque; portable; singular-named; improper-named
description: The diamonds glitter dangerously.
printed name: diamonds
```

Much of this can be seen, and seen more easily, in the World tab of the Index panel: but that only shows the initial state of play, whereas the SHOWME command reveals the situation in mid-story. ("Room", "supporter" and so on are kinds, of which more in Chapter 3.)

See **High-level debugging commands** for more convenient testing commands like this one

§2.8. The TEST command

The only way to thoroughly test a work of IF is to run a complete solution through it, and carefully check the resulting transcript of dialogue. The Skein and Transcript tools of the Inform application are provided for exactly this purpose.

All the same, most works of interactive fiction contain occasional vignettes, either in terms of short scenes of narrative, or in the behaviour of particular things or rooms, which we would like to test without the fuss of using the full story-level Skein tool. The examples in the documentation are like this: in almost every example, typing TEST ME puts the story through its paces.

Solutions or sequences for testing ("scripts") can be defined with sentences like so:

```
Test balloon with "get balloon / blow balloon / drop balloon".
```

This has no effect on the design itself, but ensures that when the story is played, typing "test balloon" will run through the given three commands in sequence, as if we had typed "get balloon" and then "blow balloon" and then "drop balloon".

The name for the test (balloon in this example) has to be a single word. Typing just "test" at the story prompt gives a list of all the test scripts known to the story. Test scripts can make use of each other, for instance:

```
Test all with "test balloon / test door".
```

One convenient way to keep track of the solution for a work being written is to include a test script at the end of each section, and to place a master test script (like "test all") at the top of the source. But different designers will prefer different approaches, and this testing system is no more than an optional convenience.

Many tests will only be sensible in given places, which may be hard to reach from the initial position; or with the aid of given things, which may be difficult to obtain. We are therefore allowed to add stipulations to test scripts:

```
Test balloon with "get balloon / blow balloon / drop balloon" holding
the balloon.
```

The "... holding the balloon" means that the balloon will be transferred to the player's ownership immediately before the test script is run, unless it is already held. Similarly:

```
Test jam with "get jam / taste jam / eat jam" in the Kitchen.
```

Or we might want to say both:

```
Test jam with "get jam / taste jam / eat jam" in the Kitchen holding the
jam.
```

(Single quotation marks in test scripts are interpreted the same way in test scripts as they are in other text: that is, they are sometimes read as double-quotes unless they appear to be present as apostrophes. The notation ['] forces a single quotation mark if necessary. Similarly, [/] forces a literal forward slash, and prevents the / from being read as dividing up two commands.)

Sometimes when testing it's convenient to get hold of something not easily available at the moment. The testing command "PURLOIN" does this:

```
The jewelled Turkish clockwork hat is in the sealed glass box.


> PURLOIN HAT
[Purloined.]
```

This can also make test scripts shorter, but of course it's important to make sure that people without PURLOIN powers can still play through.

Start of Chapter 2: The Source Text

Back to §2.7. The SHOWME command

Onward to §2.9. Material not for release

§2.9. Material not for release

Special testing commands, like "TEST" and "SHOWME", are automatically excluded from the story if it is exported from the Inform application using the Release button. We sometimes want to write our own for-testing-purposes-only code, though, and for this purpose we are allowed to designate whole headings as being "not for release":

```
Section 10 - Open sesame - Not for release

Universal opening is an action applying to nothing.
Understand "open sesame" as universal opening.
Carry out universal opening: now all doors are open.
Report universal opening: say "Open Sesame!"
```

Clearly we do not wish the final reader to be able to type "OPEN SESAME", so this whole heading will be disregarded in the Release version, as will any heading whose name includes "not for release".

Note that if a chapter, say, is marked as "not for release", then its subheadings (mere sections) will also not be for release. If in doubt, check the "Contents" index: if any section is "not for release" then so are all of its subheadings.

The reverse effect is produced by:

```
Section 10 - Open sesame - For release only
```

That is, it marks material included only in a Release version.

Start of Chapter 2: The Source Text

Back to §2.8. The TEST command

Onward to §2.10. Installing extensions

§2.10. Installing extensions

The original Inform of 1993 provided no special facilities for "extensions" - in effect, additional packets of rules providing extra features - but the creation and circulation of these extensions soon became a flourishing part of Inform culture. Today's Inform actively promotes sharing of such extensions, both to bring writers together and to support good practice. For the user of an extension, the advantage is clear: why go to great trouble to (say) work out how to make doors open automatically as needed, when somebody else has already perfected this? For the writer of an extension, there is the satisfaction of producing a good solution to a ticklish problem, and contributing to the public good.

Newcomers will probably not need extensions for quite some while, but there is nothing difficult about using them, so a few brief notes are worth giving here. (The final chapter of the documentation covers the writing of new extensions.)

Extensions are identified by name (say "Following People") and also by author (say "Mary Brown"). They need to be installed before they can be used, which means downloading them from the Internet. By far the easiest way to do this is to use the Public Library feature of Inform: then the application can do everything, letting us either choose individual extensions or download them en masse. But it's also possible to install extensions by hand.

When using Inform on Mac OS X, use the File menu item **Show Extensions Folder** to open the relevant folder in the Finder. Each author has a subfolder of this folder, and his or her extensions live inside it.

When using Inform on Windows, this means storing them in the folder

```
My Documents\Inform\Extensions
```

Each author has a subfolder of this folder, and his or her extensions live inside it. Our example extension should therefore be placed as:

```
My Documents\Inform\Extensions\Mary Brown\Following People.i7x
```

When using Inform on Linux, this means storing them in the folder

```
~/Inform/Extensions/
```

where "~" signifies your home folder. (This will have been created for you the first time you ran i7.) Each author has a subfolder of this folder, and his or her extensions live inside it. Our example extension should therefore be placed as:

~/Inform/Extensions/Mary Brown/Following People.i7x

When using Inform on Linux, this means storing them in the folder

~/Inform/Extensions/

where "~" signifies your home folder. (This will have been created for you the first time you ran i7.) Each author has a subfolder of this folder, and his or her extensions live inside it. Our example extension should therefore be placed as:

~/Inform/Extensions/Mary Brown/Following People.i7x

In fact, though, Inform can automatically install extensions for us: we need only select the "Install Extension..." item on the File menu.

The actual extension file should always be named with a ".i7x" suffix, meaning "I7 extension" - for instance, "Following People.i7x".

To provide an example, Emily Short's useful extension "Locksmith" is one of a small number of extensions which come ready-installed as part of the basic Inform package, and need not be downloaded and installed.

Each time that Inform translates any source text, it performs a quick check of the extensions available, and updates its own internal records. A directory of the extensions currently installed can be found by clicking on "Installed Extensions" from the Extensions panel. This is also worth visiting in order to browse the Public Library, a selection of extensions contributed by Inform users.

Start of Chapter 2: The Source Text

Back to §2.9. Material not for release

Onward to §2.11. Including extensions

§2.11. Including extensions

We talk about "including" such an extension into a work of IF because the process merges rules and behaviours from the extension with those we have described ourselves. It's not uncommon for contributions by five or six different people to be pooled together this way.

Including an extension is only a matter of writing a single sentence in the source. For instance:

```
Include Locksmith by Emily Short.
```

Note that it is compulsory to name both extension and author.

Many extensions come with their own documentation. Again, follow the "Installed Extensions" link to see what's available from them.

Start of Chapter 2: The Source Text

Back to §2.10. Installing extensions

Onward to §2.12. Use options

§2.12. Use options

One more preliminary. Inform has a small number of optional settings which affect the result of translating the source. The sentence:

```
Use American dialect.
```

makes the resulting work of IF use American spellings (except where the designer spells otherwise) and the American convention for spelling out numbers (thus, "one hundred seventeen" not "one hundred and seventeen"). Similarly:

```
Use the serial comma.
```

uses a comma when printing lists: thus "Julian, Dick, George, and Anne" rather than "Julian, Dick, George and Anne". A more profound change is made by

```
Use scoring.
```

which introduces the concept of a numerical score - something which modern authors of interactive fiction often feel is inappropriate, which is why Inform only provides it on

request. Two alternative options:

```
Use full-length room descriptions.
Use abbreviated room descriptions.
```

change the normal way room descriptions are shown: normally they are given in full, but in abbreviated mode, they're never given. (The latter is a bad idea in any publicly released story, but is provided for completeness and in case it may help testing.) Alternatively, we can set the traditional Infocom-style of room description to any of VERBOSE, BRIEF and SUPERBRIEF:

```
Use VERBOSE room descriptions.
Use BRIEF room descriptions.
Use SUPERBRIEF room descriptions.
```

The default is now VERBOSE, but until 2010 it was BRIEF.

Next we have:

```
Use undo prevention.
```

which disables the UNDO verb, both in play and after death, for the benefit of stories which are heavily randomised and where we do not want players to keep on UNDOing until they get a random outcome which is to their taste. (Many players consider UNDO to be their birthright, and that any work using this option is an abomination: indeed, it has even been suggested that this section of the Inform documentation be censored. To use the option is to court controversy if not outright hostility.)

We can combine any number of options in a single "Use" sentence, so for example:

```
Use American dialect and the serial comma.
```

brings about both of these changes.

Start of Chapter 2: The Source Text

Back to §2.11. Including extensions

§2.13. Administering classroom use

Inform is increasingly used in education, where teachers sometimes need to install it on a whole room of computers at once, and want to monitor their students' progress. There is no special "classroom" version of Inform, but a couple of small administration features in the standard Inform - usually never needed - might be helpful to teachers.

When Inform starts up, it now looks for a file called Options.txt inside the user's home folder for Inform. (On Mac OS X, this is "~/Library/Inform"; on Windows, "My Documents\Inform", and so on.) If the file is present, then the text in it is added to the source text of everything Inform translates.

This must be used only to set use options, specify test commands, and give release instructions. For example, the following is a valid "Options.txt":

```
Use American dialect.
Test fish with "fish/fish with pole/angle".
Release along with source text.
```

The idea is that this file can be used for setting up a standard configuration on multiple machines in a classroom setting. Here the instructor can make sure the Release button will do what she would like, and can arrange for each student's copy of Inform to respond to given Test commands: for instance, if the class has an assignment to create a simulation of a camera, the instructor could set up "Options.txt" so that TEST CAMERA would run through some commands the camera ought to respond to.

A new use option, "Use telemetry recordings.", causes Inform to copy its outcome and problem messages to files in its home folder (see above) as they occur. These files are dated, so that for instance

```
Telemetry 2009-03-25.txt
```

contains all of the recorded activity on 25 March 2009. Telemetry only records the contents of the "Problems" panel - notes of success or failure, and problem messages - and nothing is transmitted via any network, so it isn't really surveillance. The user can deliberately add a note to the current telemetry file by writing something like this in source text:

```
* "I don't get it! What's a kind? Why can't the lamp be lighted?"
```

(This is a way to make a note for the benefit of someone who will read the telemetry file - for instance, to comment on a problem message that has just appeared. Note the double-quotes. Otherwise, it's meant to look like the standard way that beta-testers mark up IF transcripts.)

These two features have been added in response to requests from education users. Let's suppose that Mr Lebling, who teaches 5th grade in Minnesota, wants to set things up just right for his class. He installs Inform on the ten computers they will use, and also copies an Options.txt file from his memory stick onto each one. The Options.txt file reads:

```
Use serial comma.
Use American dialect.
Use telemetry recordings.
```

Now Mr Lebling's class won't be confronted with English spellings, and so on. And most of the kids are happy, but Mr Lebling gets the feeling that young Marc wasn't really paying attention, so after class he checks that day's Telemetry file for that computer to see what Marc was up to, and whether he was stuck on something.

Start of Chapter 2: The Source Text

Back to §2.12. Use options

Onward to §2.14. Limits and the Settings panel

§2.14. Limits and the Settings panel

No computer has unlimited capacity, and a large, complex project may eventually bump its head against the ceiling.

Inform is a system for translating textual descriptions of interactive fiction into "story files". No single format of story file is standard to the IF community. The formats developed over the history of IF differ in three key respects:

```
- the range of computers or devices capable of playing them;
- how large they are, that is, how much play they can express;
```

```
- what extra-textual effects they can bring off.
```

Inform can write to two different formats. Neither of these is proprietary, and neither was created by the authors of Inform: each format is a community property, defined by published standards documents. An individual Inform project can make its own choice of story file format, using that project's Settings panel.

Newly created projects are set up with the Glulx format. This has largely taken over from an earlier format called the Z-machine, but Inform can still generate a version 8 Z-machine file (a so-called "z8") if required. The Z-machine is of historic importance, and may continue to be useful for certain tasks where Glulx support is not yet available, but most users will want to keep the Glulx format set all of the time.

Internally, the Inform application uses a tool called Inform 6 (which was once the entire Inform system) to manufacture the story file. There are therefore two ways that large projects can run out of space:

```
(a) By exceeding some maximum in Inform 6, or
(b) By exceeding some fundamental limitation of the current story file
format.
```

In both cases, the Inform application will display a Problems page explaining that the Inform 6 tool has failed to work as intended, and refer us to the "console output" - the text produced by Inform 6 - which is normally ignored, but can be found on the Console tab of the Results panel.

In case (a), Inform 6 will say that a memory setting has been exceeded: it will say what this setting is called (for instance "MAX_ZCODE_SIZE") and what its current value is (for instance 50000). We can then avoid the problem by adding the following use option into the source text:

```
Use MAX_ZCODE_SIZE of 60000.
```

And similarly for every other Inform 6 memory setting. (If the source tries to specify the same setting more than once - which is quite possible if extensions are included, with rival ideas - then the highest value is used.)

Case (b) is only likely to happen with the Z-machine format, since Glulx has a huge capacity; so the cure here is to switch to Glulx in the Settings. But if that's not possible for some reason - say, if we want a story file playable on a tiny handheld computer unable to manage Glulx - we still have a few options. Unless the story is very large (in which case there is little we can do), the "z8" format is most likely to be exhausted for lack of what is called "readable memory", with a message like so:

```
This program has overflowed the maximum readable-memory size of
the Z-machine format. See the memory map below: the start of the
area marked "above readable memory" must be brought down to
$10000 or less.
```

followed by a tabulation of how the Z-machine's storage has been used, a large but not very useful diagram. The first time one runs into the problem on a large project, it can be postponed, by adding the following to the source:

```
Use memory economy.
```

(Economy cuts down the verbosity of some of the testing commands, but otherwise subtracts no performance.) Writing this into the source is the equivalent of a diver switching to an emergency oxygen tank: it gives us a generous safety margin, but also tells us that now is the time to wrap things up.

If we hit the problem again, genuine cuts must be made. As a general rule, the most memory-expensive ingredients of an Inform design are various-to-various relations between large kinds such as "thing" or, if there are many rooms, "room". Other than that, if a kind has been festooned with new properties and we have created dozens of items of that kind, then we can get a fairly large saving simply by doing without one of those properties; and so on.

The ultimate memory-saving device, of course, is the one used by book publishers when there are too many pages to bind: to cut the design into two stories, Part I and Part II.

Start of Chapter 2: The Source Text

Back to §2.13. Administering classroom use

Onward to §2.15. What to do about a bug

§2.15. What to do about a bug

In its present guise, Inform is a young piece of software, and bugs are to be expected from time to time. The most obvious bugs are the ones which Inform catches itself, when it confesses that it has halted in failure, or translated the source text into a program which cannot be compiled further. But sometimes it will also happen that Inform will issue a misleading Problem message, or appear to work normally but to produce a story which does not do what it should have done.

It is very helpful for users to report faults, so that the program can be improved for everyone else. To report a fault, please first check with the Inform home page to make sure that the version of Inform you have used to detect the fault is the latest version available. You can find the latest versions at

http://inform7.com/download/

If the bug is still present in the latest version, please report the bug using Inform's bug tracking database, found at

http://inform7.com/mantis/

We can search existing bug reports using the search box at

http://inform7.com/mantis/view_all_bug_page.php

It may be that someone else has already identified the bug and even that a workaround for users is suggested. If not, please make an account at the bug tracking system and submit the requested information to help Inform's maintainers track and fix the fault.

Start of Chapter 2: The Source Text
Back to §2.14. Limits and the Settings panel
Onward to §2.16. Does Inform really understand English?

§2.16. Does Inform really understand English?

No. No computer does, and Inform does not even try to read the whole wide range of text: it is a practical tool for a particular purpose, and it deals only with certain forms of sentence useful to that purpose. Inform source text may look like "natural language", the language we find natural among ourselves, but in the end it is a computer programming language. Many things which seem reasonable to the human reader are not understood by Inform. For instance, Inform understands

```
something which is carried by the player
```

but not (at present, anyway)

```
something which the player carries
```

even though both are perfectly good English. So it is not always safe to assume that Inform will understand any reasonable instruction it is given: when in doubt, we must go back to the manual.

More philosophically, to "understand" involves contextual knowledge. Just because Inform recognises and acts on a sentence, does it really understand what we meant? It will turn out that Inform is both good and bad at this. For instance, from

```
Mr Darcy wears a top hat.
```

Inform will correctly deduce that Darcy is a person, because inanimate objects do not ordinarily wear clothes, and that the top hat is clothing. But it will not automatically know that Darcy is a man rather than a woman because it does not know the social convention implied by "Mr". Moreover, if instead we had written

```
Mr Darcy carries a top hat.
```

then Inform would not guess that the top hat is clothing. This is because it does not have the vast vocabulary and experience of a human reader: it is probably discovering the word "hat" for the first time.

Finally, it is best to avoid ambiguities rather than rely on Inform to know which meaning is patently absurd. For instance, in

```
Heatwave bone breaks clog hospital.
```

(a headline once printed by the *Oxford Mail* newspaper) a human reader quickly realises that there is no clog hospital being broken. But if Inform had been taught the verbs *to break* and *to clog* then that is exactly the conclusion it would have drawn. Or an example which genuinely arose in beta-testing:

```
The life support unit fits the egg.
```

in which Inform construed the verb as *support* and not *fits*, and then created items called "the life" (plural) and "unit fits the egg".

That disclaimer completes the groundwork, and we are ready to begin on simulating a world to explore.

Start of Chapter 2: The Source Text

Back to §2.15. What to do about a bug

Onward to Chapter 3: Things: §3.1. Descriptions

Chapter 3: Things

oors; §3.13. Locks and keys; §3.14. Devices and descriptions; §3.15. Light and darkness; §3.16. Vehicl

Contents of *Writing with Inform*
Chapter 2: The Source Text
Chapter 4: Kinds
Indexes of the examples

§3.1. Descriptions

At its simplest, the interactive fiction will be simulating a physical world to explore. The forerunner of today's IF is generally agreed to be a computer simulation by Will Crowther of the exploration of a cave system in the Mammoth and Flint Ridge chain of caves in Kentucky, a part of which might be described in Inform thus:

```
"Cave Entrance"

The Cobble Crawl is a room. "You are crawling over cobbles in a low
passage. There is a dim light at the east end of the passage."

A wicker cage is here. "There is a small wicker cage discarded
nearby."

The Debris Room is west of the Crawl. "You are in a debris room
filled with stuff washed in from the surface. A low wide passage with
cobbles becomes plugged with mud and debris here, but an awkward
canyon leads upward and west. A note on the wall says, 'Magic word
XYZZY'."

The black rod is here. "A three foot black rod with a rusty star on one
```

```
end lies nearby."

    Above the Debris Room is the Sloping E/W Canyon. West of the
    Canyon is the Orange River Chamber.
```

Here we sketch in four of Crowther's locations, and two objects: just enough to be able to walk around the caves and pick up the rod and the cage. The text in quotation marks will appear verbatim as paragraphs shown to the player as the caves are explored. The first paragraph, as we have seen, is the title of the work. The other quotations describe the places and objects introduced.

If we play this story, we find that we can type TAKE CAGE or TAKE WICKER CAGE, for instance, but not TAKE SMALL CAGE. Inform saw that we called this "a wicker cage" when it first appeared in the source text, and assumed that the player would call it that, too. (Whereas it didn't look inside the descriptive text to allow for TAKE SMALL CAGE or TAKE DISCARDED CAGE or TAKE NEARBY CAGE.) A small limitation here is that probably only the first 9 letters of each word are read from the player's command. This is plenty for handling the wicker cage and the black rod, but it might be embarrassing at a meeting of the Justice League to find that KISS SUPERHERO and KISS SUPERHEROINE read as if they are the same command.

So we have already found that Inform has made some assumptions about what we want, and imposed some limitations on how much computational effort to go to when the work of IF is finally played. If Inform guesses what we need wrongly, we need to know more advanced features of the language in order to overcome these problems. (We shall see how to change the way the player's commands are read in the chapter on Understanding.)

This is often how Inform works: make the standard way of doing things as simple as possible to describe, but allow almost any behaviour to be altered by more elaborate source text. As an example of that, the player begins in the Cobble Crawl because it was the first room created in the source text, but we could instead have written text like:

```
    The player is in the Cobble Crawl.
```

to override that. This can make the source text easier to follow if the rooms are sometimes being created in a less obvious way. For example, if we write:

```
    The silver bars are in the Y2 Rock Room.
    The Cobble Crawl is a room. South of the Crawl is Y2.
```

then the first room to be created will actually be the Y2 Rock Room, so that's where

the player will be starting unless we say otherwise.

Start of Chapter 3: Things

Back to Chapter 2: The Source Text: §2.16. Does Inform really understand English?

Onward to §3.2. Rooms and the map

Example 2: ★ **Bic** Testing to make sure that all objects have been given descriptions.

Example 3: ★ **Verbosity 1** Making rooms give brief room descriptions when revisited.

Example 4: ★★ **Slightly Wrong** A room whose description changes slightly after our first visit there.

§3.2. Rooms and the map

Rooms are joined together at their edges by "map connections", most of which are pathways in one of the eight cardinal compass directions: north, northeast (written without a hyphen), east, southeast, south, southwest, west, northwest. We also have up and down, suitable for staircases or ladders. In real life, people are seldom conscious of their compass bearing when walking around buildings, but it makes a concise and unconfusing way for the player to say where to go next, so is generally accepted as a convention of the genre.

Two more directions are provided by Inform: "inside" and "outside". These are best used when one location is, say, a meadow and the other is a woodcutter's hut in the middle of it; we might then say

```
Inside from the Meadow is the woodcutter's hut.
```

The "from" is important, as it clarifies that we intend to link two different locations, not to create an item - the hut - in a single location - the meadow.

A problem which sometimes arises when laying out maps is that Inform allows short

forms of room names to be used as abbreviations. This is usually a good idea, but has unfortunate results if we write:

```
The Airport Road is west of the Fish Packing Plant. The Airport is
west of the Airport Road.
```

...because "Airport" is taken as a reference to "Airport Road", so Inform makes only two locations, one of which supernaturally leads to itself. We can avoid this by writing:

```
The Airport Road is west of the Fish Packing Plant. A room called the
Airport is west of the Airport Road.
```

Using "called" is often a good way to specify something whose name might give rise to confusion otherwise. It always makes something new, and it is also neatly concise, because we can establish something's kind and name in the same sentence. As another example, suppose we want to create a room called "South of the Hut", to south of the Hut. We can't do so like this:

```
South of the Hut is a room. South of the Hut is south of the Hut.
```

...because Inform will read that first sentence as placing a (nameless) room to the south of a room called "Hut". Once again "called" can save the day:

```
South of the Hut is a room called South of the Hut.
```

It is best to use "called" in the simplest way possible, and in particular, best not to use "called" twice in the same sentence. Consider:

```
The kitchen cabinet contains a container called a mixing bowl and a
portable supporter called a platter.
```

It is unlikely that anyone would want to name something "a mixing bowl and a portable supporter called a platter", but not impossible, and Inform tends not to be a good judge of what is likely.

(If we really want to get rid of this issue once and for all, starting the source text with the use option "Use unabbreviated object names." will do it, but the effect is drastic. This instructs Inform not to recognise names other than in full. For example:

West of the Kitchen is the Roaring Range. South of the Range is the Pantry.

is ordinarily read by Inform as constructing three rooms (Kitchen, Roaring Range, Pantry); but with this use option set, it makes four (Kitchen, Roaring Range, Range, Pantry), in two disconnected pieces of map. Handle with care.)

Start of Chapter 3: Things

Back to §3.1. Descriptions

Onward to §3.3. One-way connections

Example 5: ★ **Port Royal 1** A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

Example 6: ★★ **Up and Up** Adding a short message as the player approaches a room, before the room description itself appears.

Example 7: ★★★ **Starry Void** Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

---

§3.3. One-way connections

Connections are ordinarily two-way, but do not have to be. One of the map connections in the Mammoth Cave simulation was made by the sentence:

The Debris Room is west of the Crawl.

Besides reading this sentence at face value, Inform also deduced that the Crawl was probably meant to be east of the Debris Room: in other words, that the path between

them is a two-way one. When Inform makes guesses like this, it treats them as being less certain than anything explicitly stated in the source. Inform will quietly overturn its assumption if information comes to hand which shows that it was wrong. That might happen in this case if another sentence read:

```
The Hidden Alcove is east of the Debris Room.
```

These two sentences are not contradictory: Inform allows them both, simply accepting that the world is more complicated than it first assumed. There are relatively few situations where Inform has to make educated guesses, but when it does, it tries always to follow Occam's Razor by constructing the simplest model world consistent with the information in the Source text.

We can even explicitly make a route which turns around as it leads between two rooms:

```
West of the Garden is south of the Meadow.
```

If we want to establish a route which cannot be retraced at all, we can specify that a particular direction leads nowhere:

```
East of the Debris Room is nowhere.
```

Finally, note that Inform's assumptions about two-way directions are only applied to simple sentences. When the source text seems to be saying something complicated, Inform takes it as a precise description of what's wanted. So, for example, in:

```
The Attic is above the Parlour.
The Attic is a dark room above the Parlour.
```

Inform makes guesses about the first sentence, and makes a two-way connection; but it accepts the second sentence more precisely, with just a one-way connection.

Start of Chapter 3: Things

Back to §3.2. Rooms and the map

Onward to §3.4. Regions and the index map

Example 8: ★ **Port Royal 2**

Another part of Port Royal, with less typical map connections.

Example 9: **The Unbuttoned Elevator Affair** A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

---

§3.4. Regions and the index map

Rooms represent individual places to which one can go, but we tend to think of the world around us in larger pieces: we think of a house and a garden, rather than each of the single rooms of the house and all corners of its garden. To Inform a collection of rooms is called a "region", and we can create one like so:

```
The Arboretum is east of the Botanical Gardens. Northwest of the
Gardens is the Tropical Greenhouse.

The Public Area is a region. The Arboretum and Gardens are in the
Public Area.
```

The real usefulness of creating regions like "Public Area" will only appear later, when we begin defining rules of play which apply in some areas but not others, but in the mean time we can see the effect by turning to the World tab of the Index. In the World Index, Inform draws a map - or at least a stylised attempt at a diagram of the rooms and their connections: this will not always correspond to how we imagine things, but with any luck it should mostly be right.

Rooms are represented by coloured squares, and the colour-coding is done by region. In the above example, the two "Public Area" rooms are coloured green (as it happens); the Greenhouse, since it belongs to no region, is a neutral grey.

Regions can be put inside each other:

```
The University Parks is a region. The Public Area is in the University
Parks.
```

but they are not allowed to overlap other than by one being entirely inside the other.

See **Improving the index map** *for ways to adjust the way the index map is drawn or exported for publication*

Start of Chapter 3: Things

Back to §3.3. One-way connections

Onward to §3.5. Kinds

Example 10: **Port Royal 3** Division of Port Royal into regions.

---

§3.5. Kinds

The following description runs to only 33 words, but makes a surprisingly intricate design. It not only places things within rooms, but also places them very specifically with respect to each other:

```
"Midsummer Day"

East of the Garden is the Gazebo. Above is the Treehouse. A
billiards table is in the Gazebo. On it is a trophy cup. A starting pistol
is in the cup.
```

Inform needs to identify the places and objects being described by the nouns here, and to guess what it can about them. For instance, the pistol can be picked up but not walked inside, whereas the Treehouse is the reverse. (This is obvious to someone who knows what these words mean, less obvious to a computer which does not, but the text contains sufficient clues.) Inform does this by sorting the various nouns into different categories, which are called "kinds". For instance:

```
Garden, Gazebo, Treehouse - room
billiards table - supporter
cup - container
starting pistol - thing
East, up (implied by "above") - direction
```

(A container is something which can contain other things, and a supporter similarly.)

For instance Inform knows that if one thing is in another, then the second thing is either a room or a container, and if one thing is on another, the second thing is a supporter. This worked nicely for the design above, but:

```
In the Treehouse is a cardboard box.
```

results in the cardboard box being made only a "thing": because nothing has been put inside it, there is no reason for Inform - which does not know what a cardboard box looks like - to guess that it is a "container". So we need to add:

```
The box is a container.
```

It is rather clumsy to have to write two sentences like this, so we would normally write this instead:

```
In the Treehouse is a container called the cardboard box.
```

Start of Chapter 3: Things

Back to §3.4. Regions and the index map

Onward to §3.6. Either/or properties

Example 11: ★ **First Name Basis** Allowing the player to use different synonyms to refer to something.

Example 12: ★ **Midsummer Day** A few sentences laying out a garden together with some things which might be found in it.

§3.6. Either/or properties

Some containers, like bottles, can be opened: others, like buckets, cannot. If they can be opened, then sometimes they will be open, and sometimes closed. These are examples of properties, which can change during play. The following source sets

some properties:

> The cardboard box is a closed container. The glass bottle is a transparent open container. The box is fixed in place and openable.

There are only four different properties referred to here. Closed means not open, and vice versa, so these two adjectives both refer to the same property. (As might be expected, when a container is open, one can see inside and place things within, or take them out.) The glass bottle and the box being containers is a matter of their kinds, which is something fundamental and immutable, so "container" does not count as a property.

A "transparent" container is one which we can see inside even when it is closed, and the opposite is an "opaque" container.

The property of being "fixed in place" ensures that the player cannot pick the item up and walk away with it: this is useful for such things as oak trees or heavy furniture. The opposite condition is to be "portable".

A container which is "openable" can be opened or closed by the player; as might be expected, the opposite is "unopenable".

With a really large cardboard box, we might imagine that the player could get inside: such a container should be declared "enterable".

Start of Chapter 3: Things

Back to §3.5. Kinds

Onward to §3.7. Properties depend on kind

Example 13: ★ **Tamed**
Examples of a container and a supporter that can be entered, as well as nested rooms.

§3.7. Properties depend on kind

Properties depend very much on kind. It makes no sense to ask whether a room is transparent or opaque, for instance, so Inform will not allow this either to be specified or queried.

Another way that kind influences properties can be seen from an earlier example:

```
The Gazebo is a room. A billiards table is in the Gazebo. On it is a
trophy cup. A starting pistol is in the cup.
```

The cup, the pistol and the table are all allowed to have the "fixed in place" property, but in fact only the table actually has it: the cup and the pistol are created as "portable" instead. This is because Inform knows that most things are portable, but that supporters - such as the table - are usually fixed in place. If this assumption is wrong, we need only add the line:

```
The table is portable.
```

Start of Chapter 3: Things

Back to §3.6. Either/or properties

Onward to §3.8. Scenery

Example 14: **Disenchantment Bay 1** A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

§3.8. Scenery

As we have just seen, making something "fixed in place" will prevent it from being picked up or moved. But it remains substantial enough to be described in its own paragraph of text when the player visits its location. This can be unfortunate if it has also been described already in the body of the main description for that location. For instance, if we wrote:

```
The Orchard is a room. "Within this quadrille of pear trees, a single
gnarled old oak remains as a memory of centuries past." The gnarled
old oak tree is fixed in place in the Orchard.
```

This would end up describing the oak twice, once in the paragraph about the Orchard, then again in a list of things within it:

```
Orchard
Within this quadrille of pear trees, a single gnarled old oak remains
as a memory of centuries past.


You can see a gnarled old oak tree here.
```

We avoid this by making it "scenery" instead of "fixed in place":

```
The gnarled old oak tree is scenery in the Orchard.
```

Any thing can be scenery, and this does not bar it from playing a part in the story: it simply means that it will be immobile and that it will not be described independently of its room. Being immobile, scenery should not be used for portable objects that are meant to be left out of the room description.

If a supporter is scenery, it may still be mentioned in the room description after all, but only as part of a paragraph about other items, such as

```
On the teak table are a candlestick and a copy of the Financial
Times.
```

If the player takes the candlestick and the Times, the teak table will disappear from mention. (Scenery containers do not behave in this way: their contents are assumed to be less immediately visible, and will be mentioned only if the player looks inside them.)

Start of Chapter 3: Things

Back to §3.7. Properties depend on kind

Onward to §3.9. Backdrops

Example 15: ★ **Disenchantment Bay 2** Disenchantment Bay: creating some of the objects in the cabin's description.

Example 16: ★ **Replanting** Changing the response when the player tries to take something that is scenery.

§3.9. Backdrops

It is a cardinal rule that nothing can be in more than one place at the same time, but rules were made to be broken, and an exception is allowed for a special kind of thing called a "backdrop". For instance:

```
"Streaming"

The Upper Cave is above the Rock Pool.

The stream is a backdrop. It is in the Upper Cave and the Rock Pool.
```

Backdrops are ordinarily in the background: if the sky needed to be referred to in the course of play, it might be represented by a backdrop, for instance. Here we have a stream of water running through two rooms, though it might be any number. Backdrops are always fixed in place.

Backdrops can be put in regions as well as rooms, and if so, then they are present at every room in the given region (or regions), as well as any specific rooms they may also be put into. For instance:

```
The Outdoors Area is a region. The Moon is a backdrop. The Moon is
in the Outdoors Area. The Moon is in the Skylight Room.
```

The special place "everywhere" can be given as the location of a backdrop to make it omnipresent:

```
The sky is a backdrop. The sky is everywhere.
```

Inform assumes that backdrops are also scenery unless told otherwise, so this will not result in messages like "You can also see the sky here." being included in room descriptions. In the case of the stream above, we could artfully mention it in passing in the room descriptions of the Upper Cave and the Rock Pool.

See **Moving backdrops** for ways to place backdrops in dynamically changing selections of rooms

Start of Chapter 3: Things

Back to §3.8. Scenery

Onward to §3.10. Properties holding text

Example 17: **Disenchantment Bay 3** Disenchantment Bay: adding a view of the glacier.

§3.10. Properties holding text

The properties we have seen so far have all been either/or: either open or closed, either transparent or opaque, either fixed in place or portable, either openable or not openable. However, some properties can have a much wider range of possibilities. For instance, the "description" of a room is the text revealed when the player first enters it, or types "look". This needs to be textual: Inform would complain if, for instance, we tried to set the description of something to the number 42. We have already seen a concise way to set the description of a room:

```
The Painted Room is north of the Undertomb. "This is the Painted
Room, where strange wall drawings leap out of the dark at the gleam
of your candle: men with long wings and great eyes, serene and
morose."
```

This does the same thing as:

```
The Painted Room is north of the Undertomb. The description of the
Painted Room is "This is the Painted Room, where strange wall
drawings leap out of the dark at the gleam of your candle: men with
long wings and great eyes, serene and morose."
```

Or even:

```
The Painted Room is north of the Undertomb. The description is
"This is the Painted Room, where strange wall drawings leap out of
```

the dark at the gleam of your candle: men with long wings and great eyes, serene and morose."

Start of Chapter 3: Things
Back to §3.9. Backdrops
Onward to §3.11. Two descriptions of things

§3.11. Two descriptions of things

The player's first sight of something is the text used as its "initial appearance":

> The plain ring is here. "Cast aside, as if worthless, is a plain brass ring."

This text appears as a separate paragraph in the text describing the Painted Room. It will continue to be used until the first time player picks the ring up (if this ever happens), so it normally describes things in their original, undisturbed context. (Inform uses an either/or property called "handled" for this: something is "handled" if it has at some point been held by the player.)

Thus when a piece of text stands alone as a sentence in its own right, then this is either the "description" of the most recently discussed room, or the "initial appearance" of the most recently discussed thing. Either way, it is used verbatim as a paragraph in the text shown to the player visiting the room in question.

But a thing also has an ordinary "description", which is used to give a close-up look at it. This text is ordinarily only revealed to the player when a command like "examine ring" is keyed in:

> The description of the plain ring is "No better than the loops of metal the old women use for fastening curtains."

See **Creating a scene** for the description of a scene, which is set in the same way

Start of Chapter 3: Things
Back to §3.10. Properties holding

text

Onward to §3.12. Doors

Example 18: ★ **Disenchantment Bay 4** Disenchantment Bay: fleshing out the descriptions of things on the boat.

Example 19: ★★ **Laura** Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

---

§3.12. Doors

The map of an interactive fiction is the layout of rooms and the entrances and exits which connect them. So far, these map connections have always run from one room to another, like so:

```
The Painted Room is north of the Undertomb.
```

However, we can also interpose doors between rooms, like so:

```
The heavy iron grating is east of the Orchard and west of the
Undertomb. The grating is a door.
```

The second sentence is needed since otherwise Inform will take "heavy iron grating" to be the name of a third room, whereas what we want is for the grating to be something physically present in both the Orchard and in the Undertomb, and acting as a conduit between them. To this end it needs to be a "door", a kind we have not so far seen. In the absence of any other instruction, a newly created door will be fixed in place, closed and openable.

The grating really does come in between the two rooms: the grating is what lies immediately east of the Orchard, not the Undertomb room. So if we wrote the following:

```
The Undertomb is east of the Orchard. The heavy iron grating is east
```

```
of the Orchard and west of the Undertomb. The grating is a door.
```

then Inform would say that this is a contradiction: we said the Undertomb was east of the Orchard, but then we said that the grating was east of the Orchard.

Inform's "door" kind can be used for all manner of conduits, so the word door need not be taken literally. In Ursula K. Le Guin's beguiling novel "The Tombs of Atuan", from which the above rooms are stolen, it is not a grating which interposes, but:

```
The red rock stair is east of the Orchard and above the Undertomb.
The stair is an open door. The stair is not openable.
```

In real life, most doors are two-sided, and can be used from either of the rooms which they join, but this is not always convenient for interactive fiction. Here is a one-sided door:

```
The blue door is a door. It is south of Notting Hill. Through it is the
Flat Landing.
```

(Note the use of "it" here as an optional abbreviation.) This will make a door visible only on the Notting Hill side; no map connection will be made in the reverse direction, unless we ask for one.

So much for creating and describing individual doors. Once we need to write about doors in general, we are likely to want a way to find out where a given door sits in the map. The following phrases reveal this:

front side of (object) **... room**

This phrase produces the first of the one or two rooms containing a door - first in the order given in the source text. Example: if

```
The red rock stair is east of the Orchard and above the
Undertomb.
```

then "front side of the red rock stair" produces the Orchard. For a one-sided door, this produces the only room containing the door.

back side of (object) **... room**

This phrase produces the last of the one or two rooms containing a door - last in the order given in the source text. Example: if

> The red rock stair is east of the Orchard and above the Undertomb.

then "back side of the red rock stair" produces the Undertomb. A one-sided door has no "back side."

More often, we are dealing with a door and want to know what it leads to, but that depends where we're standing:

other side of (door) **from** (room) **... object**

This phrase produces the room on the other side of the door, as seen from the given vantage point, which needs to be one of its sides. Example: if

> The red rock stair is east of the Orchard and above the Undertomb.

then "other side of the red rock stair from the Undertomb" produces the Orchard, and vice versa.

direction of (door) **from** (room) **... object**

This phrase produces the direction in which the door leads, as seen from the given vantage point, which needs to be one of its sides. Example: if

> The red rock stair is east of the Orchard and above the Undertomb.

then "direction of the red rock stair from the Undertomb" produces up.

See **Adjacent rooms and routes through the map** for more phrases which can look at the current map layout

---

Start of Chapter 3: Things
Back to §3.11. Two descriptions of things
Onward to §3.13. Locks and keys
Example 20: ★ **Disenchantment Bay 5** Disenchantment Bay: adding the door and the deck to our charter boat.
Example 21: ★★ **Escape** Window that can be climbed through or looked through.
Example 22: ★★★ **Garibaldi 1** Providing a security readout device by which the player can check on the status of all doors in the game.

§3.13. Locks and keys

It seems unwise for a door in Notting Hill to be unlocked, so:

> The blue door is lockable and locked. The matching key of the blue door is the brass Yale key.

Since the second sentence here is a little clumsy, we can equivalently say

> The brass Yale key unlocks the blue door.

Yet a third way to say this is:

> The blue door has matching key the brass Yale key.

This introduces three new properties: a door can be locked or unlocked; lockable or not lockable; and it can have a matching key, which must be another thing. The same thing can be the matching key of many different locks: and note that a door can be

locked and even lockable without having a matching key at all, in which case the player trying to open it will be permanently out of luck. Doors are ordinarily unlocked, not lockable, and without a matching key.

Containers can also have locks, in exactly the same way, and are allowed to have the same properties. On the other hand supporters never have locks: it makes no sense to be able to lock a tabletop, for instance, and Inform will not allow any discussion of the matching key of a supporter, or of a supporter being locked or unlocked.

Start of Chapter 3: Things

Back to §3.12. Doors

Onward to §3.14. Devices and descriptions

Example 23: ★ **Disenchantment Bay 6** Disenchantment Bay: locking up the charter boat's fishing rods.

Example 24: ★★ **Neighborhood Watch** A locked door that can be locked or unlocked without a key from one side, but not from the other.

---

§3.14. Devices and descriptions

A "device" is another of the standard kinds of thing, and should be used for anything which can be switched on or off: a light switch, say, or a slide projector. Devices are generally machines, clockwork or electrical. A device is always either "switched on" or "switched off", but is switched off unless we specify otherwise.

That makes three kinds of thing which will likely change their appearance according to which of their two possible states they are in: doors and containers, which can be open or closed; and devices, which can be switched on or switched off. We would like to produce text accordingly, and we can do this using Inform's ability to make (almost) any piece of text change with circumstances. For instance:

> The coffin is an openable container in the Undertomb. "[if open]The lid of a plank coffin yawns open.[otherwise]A plank coffin lies upon the dirt floor of the Tomb."

We could use a similar trick to make the appearance of a device change "if switched on". There will be much more about text substitutions, as instructions in square brackets like these are called, in later chapters.

See **Text with substitutions** *for more on varying what is printed*

Start of Chapter 3: Things

Back to §3.13. Locks and keys

Onward to §3.15. Light and darkness

Example 25: ★ **Disenchantment Bay 7** Disenchantment Bay: making the radar and instruments switch on and off.

Example 26: ★★ **Down Below** A light switch which makes the room it is in dark or light.

§3.15. Light and darkness

Rooms can be "dark" or "lighted", though they are lighted by default, and are lighted in all the examples we have seen so far.

> The Sinister Cave is a dark room. "A profoundly disquieting rock formation, apparently sculptured by some demonic hand, this is not a cave in which to relax."

When the player is in a dark room, he can still go in various directions, but he cannot see the room description or interact with any of the objects in the room, except those he is holding. This means that, unless we should change the Cave in some way during play, the text above ("A profoundly...") will only be read if the player succeeds in bringing light into the Cave, perhaps by bringing along the following:

> The flaming torch is in the Sandy Passage. "Stuck loosely into the sand is a flaming torch." The flaming torch is lit.

A thing with the property of being "lit" will enable the player to see inside dark rooms,

and to carry out other activities requiring light, such as examining items. A lit thing in an open container will still light up a room; a lit thing in a closed container will not, unless the container has been given the "transparent" property.

It is possible to adjust the way darkness behaves, and we will see more on this topic in the chapter on Activities.

See **Printing a refusal to act in the dark** *for the first of several ways to control what is printed in the dark*

Start of Chapter 3: Things

Back to §3.14. Devices and descriptions

Onward to §3.16. Vehicles and pushable things

§3.16. Vehicles and pushable things

Next in the tour of standard kinds is the "vehicle". This behaves like (indeed, is) an enterable container, except that it will not be portable unless this is specified.

```
In the Garage is a vehicle called the red sports car.
```

The player can enter the sports car and then move around riding inside it, by typing directions exactly as if on foot: and the story will print names of rooms with "(in the red sports car)" appended, lest this be forgotten.

We have already seen that some things are portable, others fixed in place. In fact we can also make a third sort of thing: those which, although not portable, can be pushed from one room to another with commands like "push the wheelbarrow north". At a pinch, we might just be willing to allow:

```
The red sports car is pushable between rooms.
```

But of course this is a property which almost any thing can have, not just a vehicle. (Only "almost" because Inform will not allow a door to be pushable between rooms, in the interests of realism rather than surrealism.)

If we need vehicles which the passenger sits on top of, like a horse or a tractor, the standard "vehicle" kind will not be ideal. However, by loading one of the extensions which comes ready-installed:

```
Include Rideable Vehicles by Graham Nelson.
```

...we are provided with two more kinds, "rideable vehicle" and "rideable animal", just right for the tractor and the horse respectively. (As with all extensions, the documentation can be seen by clicking Go on some source which contains the above line, and then turning to the Contents index; or from the Installed Extensions tab of the Extensions panel.)

See **Going by, going through, going with** *for further ways to customize vehicle behaviour*

Start of Chapter 3: Things

Back to §3.15. Light and darkness

Onward to §3.17. Men, women and animals

Example 27: ★ **Peugeot** A journey from one room to another that requires the player to be on a vehicle.

Example 28: ★★ **Disenchantment Bay 8** Disenchantment Bay: a pushable chest of ice for the boat.

Example 29: ★★★ **Hover** Letting the player see a modified room description when he's viewing the place from inside a vehicle.

---

§3.17. Men, women and animals

Rounding out the standard kinds provided by Inform are four for living things: "person", which is a kind of thing, and "man", "woman" and "animal", all kinds of person. For instance:

```
In the Ballroom is a man called Mr Darcy.
```

For the time being, men and women will be little more than waxworks: they will come to life only when we go beyond the present stage of creating an initial state of the world.

People can be male or female: this is an either/or property for the "person" kind, and it affects play at run-time a little, because the player can use "him" and "her" to refer to male or female people encountered. Men and women are always male and female respectively, and for animals we can choose either way, for example making a stallion male or a nanny goat female. Animals are male unless we say otherwise.

If our animal is instead something like a beetle or an earthworm, where gender doesn't seem to matter or even to exist, we can use the further property "neuter":

```
The spider is a neuter animal in the Bathroom.
```

The Standard Rules don't make people behave differently according to their genders, and the main difference comes down to language: whether we want the animal to be called "her", or "it". Because of the existence of "neuter", we sometimes need to be cautious about the use of the adjective "male": since Inform, partly for historical reasons, uses an either/or property for masculinity, neuter animals are also "male".

Start of Chapter 3: Things

Back to §3.16. Vehicles and pushable things

Onward to §3.18. Articles and proper names

Example 30: ★ **Disenchantment Bay 9** Disenchantment Bay: enter the charter boat's Captain.

§3.18. Articles and proper names

Suppose we have said that:

```
In the Ballroom is a man called Mr Darcy.
```

When the Ballroom is visited, the man is listed in the description of the room as "Mr Darcy", not as "a Mr Darcy". This happened not because Inform recognised that Darcy is a proper name, or even because men tend to have proper names, but because Inform noticed that we did not use "a", "an", "the" or "some" in the sentence

which created him. The following shows most of the options:

```
The Belfry is a room. A bat is in the Belfry. The bell is in the Belfry.
Some woodworm are in the Belfry. A man called William Snelson is in
the Belfry. A woman called the sexton's wife is in the Belfry. A man
called a bellringer is in the Belfry.

In the Belfry is a man called the vicar. The indefinite article of the
vicar is "your local".
```

In the resulting story, we read:

```
You can see a bat, a bell, some woodworm, William Snelson, the
sexton's wife, a bellringer and your local vicar here.
```

The subtlest rule here is in the handling of "the". We wrote "The bell is in the Belfry", but this did not result in the bell always being called "the" bell: in fact, writing "A bell is in the Belfry" would have had the same effect. On the other hand, "A woman called the sexton's wife is in the Belfry." led to the wife always being known as "the" sexton's wife, not "a" sexton's wife, because Inform thinks the choice of article after "called" shows more of our intention than it would elsewhere. These rules will never be perfect in all situations, so we are also allowed to specify indefinite articles by hand, as the vicar's case shows.

"Some" is worth a closer look, because English uses it in several different ways. By introducing the woodworm with "some", above, we established that it was plural. We might imagine that there are many worms, even though they are represented by a single thing in Inform. We can expect to see text in the story such as:

```
You can see some woodworm here.
The woodworm are fixed in place.
```

But suppose we wanted something which there is an amount of, but which is not made up of individual items - a so-called mass noun like "water", or "bread". Now we can write:

```
The water is here. The indefinite article is "some".
```

and this time Inform does not treat the "some water" thing as a plural, so we might read:

```
You can see some water here.
The water is hardly portable.
```

rather than "The water are hardly portable."

Finally, we can override these settings, if they still come out not as we intend, by explicitly changing the either/or properties "singular-named" (vs "plural-named") and "proper-named" (vs "improper-named").

Start of Chapter 3: Things

Back to §3.17. Men, women and animals

Onward to §3.19. Carrying capacity

Example 31: ★ **Belfry** You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

Example 32: ★★ **Gopher-wood** Changing the name of a character in the middle of play, removing the article.

§3.19. Carrying capacity

The containers and supporters created so far have been boundlessly capacious: or rather, though we seldom notice the difference, have had a maximum carrying capacity of 100 items. This is clearly unrealistic for a small purse or a modest mantelpiece. We can impose upper limits with sentences like so:

```
The carrying capacity of the jewelled purse is 2.
```

```
The bijou mantelpiece has carrying capacity 3.
```

Attempts by the player to overfill, or overload, will now be rebuffed with a message such as "There is no room on the mantelpiece".

The player is not a container or a supporter, but nevertheless does have a carrying capacity: this is interpreted to mean the maximum number of items which can be carried at once.

```
The carrying capacity of the player is 4.
```

These restrictions only apply to the player (and other in-world characters): as the omnipotent creators, we are not restrained by them. Nothing prevents this:

```
The carrying capacity of the jewelled purse is 2. The diamond, the
ruby and the sapphire are in the purse.
```

The player will be able to remove all three items, but only put two of them back. (This is probably something we only want very occasionally: perhaps to create a sack stuffed almost to bursting point.)

Start of Chapter 3: Things

Back to §3.18. Articles and proper names

Onward to §3.20. Possessions and clothing

§3.20. Possessions and clothing

We have seen how to place objects in rooms, and in containers or on supporters. But what about people? Perhaps it could be said that they "contain" the fillings in their teeth, or "support" a top hat, but this is not very natural. Inform therefore never speaks of things being "in" or "on" people. Instead, they have two sorts of possessions: the things they carry, and the things they wear. (Body parts, such as arms and legs, are different again: see "parts" below for a clue to how to do these.) Thus:

```
Mr Darcy wears a top hat. Mr Darcy carries a silver sword.
```

In fact, Inform deduces from this not only who owns the hat and the sword, but also that Darcy has the kind "person", because only people can wear or carry.

As all the assertion verbs do, "to wear" and "to carry" have participles which Inform knows about. So we could equally well write:

```
The scarlet coat is worn by Mr Wickham. The duelling pistol is carried
by Mr Wickham.
```

If we do not specify who does the wearing, or carrying, then this is assumed to be the player. Thus:

```
A brass lantern and a rusty iron key are carried. The mosquito-
repellent hat is worn.
```

It would make no sense to "wear" the key, for instance, so Inform needs to distinguish between what is clothing and what is not. It does this with an either/or property called "wearable": if something has this property then the player will be allowed to wear it, provided it can first be picked up. Anything which is worn by somebody at the start of play is assumed to be wearable (unless we say otherwise). But if nobody is initially wearing the item in question, then we have to be explicit:

```
The player carries a scarlet gown. The gown is wearable.
```

(When we come to asking questions about the current situation, we will need to remember that "to carry" and "to wear" are different. Thus "if Lancelot carries the plate armour" will not be true if he is wearing it rather than carrying it under his arm. As we will later see, we can instead vaguely say "if Lancelot has the plate armour" to mean either carrying or wearing.)

See **To carry, to wear, to have** *for a more detailed explanation of carrying, wearing, and possessing as Inform understands them*

Start of Chapter 3: Things

Back to §3.19. Carrying capacity

Onward to §3.21. The player's holdall

Example 33: **Disenchantment Bay 10** Disenchantment Bay:

things for the player and the characters to wear and carry.

§3.21. The player's holdall

When the player has only limited carrying capacity, play is likely to be tiresome, but we can make life easier by providing a way for the player to carry endless items without dozens of free hands to hold them all:

```
"Sackcloth"

The Attic is a room. The old blue rucksack is a player's holdall. The
player is wearing the rucksack.

The carrying capacity of the player is 3.

In the Attic are a CD entitled No Smoke Without Fire, a 70s
photograph of an American winning Wimbledon, a fraxinus branch,
an urn holding your late great-aunt's remains, a convention badge
from the American Society of Hypertension and a ghost story by M R
James.
```

This example story introduces a new kind of container, the "player's holdall". This is a kind of which most stories will contain at most one example, but in principle there can be any number. A player's holdall is a capacious bag into which the player automatically places surplus items whenever his or her hands are full: trying the above example story and getting the items one by one will give the general idea.

Of course, if the carrying capacity of the player is never reached then there will never be any surplus items and a player's holdall will behave just like any other (portable, usually openable) container.

See **Units** for the tools to implement a more sophisticated capacity system

Start of Chapter 3: Things

Back to §3.20. Possessions and clothing
Onward to §3.22. Food

Example 34: ★ **Disenchantment Bay 11** Disenchantment Bay: making a holdall of the backpack.

§3.22. Food

We have nearly reached the end of the chapter on Things, but one either/or property for things remains: every thing is either "edible" or "inedible". Unless we say otherwise, things are inedible. But for instance we might write:

```
The player carries a Macintosh apple. The Macintosh is edible.
```

(The type of computer is named after a variety of apple descended from a tree cultivated in 1811 by John McIntosh of Ontario.) Edible things are just like inedible ones, except that the player can EAT them. This will usually only consume the foodstuff in question, effectively destroying it, but using techniques from later chapters we could make the consequences more interesting.

Start of Chapter 3: Things
Back to §3.21. The player's holdall
Onward to §3.23. Parts of things

§3.23. Parts of things

Everything has one and only one kind. This is both good and bad: good for clarity, bad if something needs to behave in two different ways at once. How might we simulate a car with an ignition key, given that no single thing can be both a "vehicle" and a "device" at the same time?

The Inform world model takes the view that such a car is too complicated to be simulated with a single thing. Instead it should be simulated as a vehicle (the car) which has a device (the ignition) attached. This is done using a third kind of containment to those seen so far ("in..." and "on..."): "part of".

"Buttons"

The Confectionary Workshop is a room. The Chocolate Machine is here. "The Chocolate Machine has pride of place. A lever and two buttons, one white, the other brown, seem to be the only controls. On top is a hopper."

A container called the hopper is part of the Chocolate Machine. The lever, the white button and the brown button are parts of the Chocolate Machine.

The Chocolatier's desk is here. "The Chocolatier evidently works at the imposing green-leather topped desk facing the Machine. It has three drawers with brass handles."

The upper drawer, the middle drawer and the lower drawer are parts of the desk. The upper drawer, the middle drawer and the lower drawer are openable closed containers. In the middle drawer is a sugared almond. In the lower drawer is a Battenburg cake. On the desk is a liquorice twist.

The cake, the twist and the almond are edible.

The machine and the desk each have several "parts" representing subsidiary pieces of themselves. The desk is a "supporter" (it needs to be, for the liquorice twist to be on top) but also has three "containers" attached, each of which can be opened or closed independently.

In the interests of realism, the standard rules of play protect these composite things. Thus if the desk were to be moved elsewhere (rolling on sugar casters perhaps) then its parts would move with it, and the player is not allowed to detach parts of things: the drawers can be opened or closed, but not pulled out altogether.

Note that rooms and regions are not allowed to have parts. (Rooms are already parts of regions, and to divide up rooms, we can either make several rooms or place containers or other obstacles in a single one.)

Start of Chapter 3: Things

Back to §3.22. Food

Onward to §3.24. Concealment

Example 35: ★★★ **Fallout Enclosure** Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

Example 36: ★★★ **Brown** A red sticky label which can be attached to anything in the game, or removed again.

Example 37: ★★★★ **Disenchantment Bay 12** A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

§3.24. Concealment

Though realism can become tiresome in interactive fiction, there are times when we cannot go along with Inform's normal assumption that all of a person's possessions are visible to everybody else. People are not like containers, which either show all of their holdings or not, according to whether they are open or transparent. If a man is carrying a fishing rod and a wallet, one will be on open show, the other not. Some clothing is outwardly visible, but not all.

Whether or not something is concealed is not like the either/or properties we have seen so far - such as being "open" or "closed" - because it is not really a property of the thing itself, but depends on the habitual behaviour of its current owner. To talk about behaviour we have to use sentences of a kind not seen so far, and which will not fully be explained for some chapters to come.

But straightforward cases are easy to write, if only by imitating the following examples.

Here we make the Cloaked Villain invariably conceal anything she is holding or wearing:

```
Rule for deciding the concealed possessions of the Cloaked Villain:
yes.
```

At which point we think about it more carefully, and then rewrite:

```
Rule for deciding the concealed possessions of the Cloaked Villain: if
the particular possession is the sable cloak, no; otherwise yes.
```

(A rule which says neither "yes" nor "no" will decide yes, but it's best to spell out exactly what's wanted.)

Parts are treated exactly as if clothes or items being held, and the following will make the face and inscription on a coin invisible unless the player is holding it - the idea being that they are too small to be seen from farther away.

```
The coin is in the Roman Villa. The face and inscription are parts of
the coin. Rule for deciding the concealed possessions of the coin: if
the coin is carried, no; otherwise yes.
```

There is also an either/or property called "described"/"undescribed", intended to be used only as a last resort, but which has the ability to hide something from room descriptions. This not really hiding: the idea is that "undescribed" should be used only for cases where some other text already reveals the item, or where its presence is implicit. Even then, it should only be used when the item is intended to be taken or moved by the player at some point - if the item isn't intended to move, it's much better to make it "scenery". (There's only one commonly-found example - the player's own body, the "yourself", is undescribed.)

Note that the "undescribed" property is automatically removed from anything carried by, worn by or part of the player, even indirectly; and that nothing on top of an "undescribed" supporter will be visible in a room description, even if it itself is "described". (Scenery supporters don't suffer from that restriction, which is one reason scenery is a better option when possible.)

Start of Chapter 3: Things

Back to §3.23. Parts of things

Onward to §3.25. The location of something

Example 38: ★★★ **Search and Seizure** A smuggler who has

§3.25. The location of something

The model world created by Inform is partitioned into rooms. This means that everything which exists in the model world, exists in one of the rooms. If we write a sentence such as

```
Professor Wilderspin is a man.
```

and say nothing more about Wilderspin, then he does not physically exist at the start of the story: he is said to be "out of play", and stays that way until we move him into one of the rooms. A better metaphor might be that he is waiting in the wings, ready to come onto the stage.

Every thing is either out of play, or can be found in one of the rooms, and the property "location of X" gives us the room in question. The following condition tests, in effect, whether Wilderspin is in play:

```
if the location of Wilderspin is a room, ...
```

Which uses a new phrase:

location of (object) **... room**

This phrase produces the room which, perhaps indirectly, contains the object given. Example: if the player stands in Biblioll College and wears a waistcoat, inside which is a fob watch, then

```
location of the fob watch
```

is Biblioll College. In general, a thing cannot be in two rooms at once, but there are two exceptions: two-sided doors, present on both sides, and backdrops. The "location of" a door is its front side, but a backdrop has no location. (Objects which are not things at all, such as rooms and directions, also have no location.)

We very often want to know the location of the player, and this is more simply called

just "the location". (This is actually a value that varies rather than a phrase, but that's a technicality we can ignore here.)

The idea of indirect containment is useful enough to have a name: Inform calls it "enclosure". A thing encloses whatever is a part of itself, or inside itself, or on top of itself, and it also encloses anything that they enclose. And when something moves around, anything it encloses will move with it. In the example above, Biblioll College (a room) and the player (a person) both enclose the fob watch and the waistcoat. (The small print: a door is enclosed by the rooms on both sides; a backdrop is never enclosed.)

Enclosure is only useful when being used as a question. So the following is fine:

```
if the player encloses the fob watch, ...
```

But these will produce problem messages:

```
The player encloses the fob watch. The location of the trilobite is the
Museum.
```

because they are too vague. Inform needs to know exactly where the fob watch and the trilobite will begin the story, whereas these sentences leave room for doubt about who or what is actually holding them.

Start of Chapter 3: Things
Back to §3.24. Concealment
Onward to §3.26. Directions

Example 39: ★★ **Van Helsing**
A character who approaches the player, then follows him from room to room.

§3.26. Directions

"Direction" is a kind which is quite unlike most of those seen so far. While it has to do with the physical world, a direction does not exactly belong to it. One cannot find "southeast" sitting on a shelf. "Direction" is not a kind of thing, nor a kind of room: it is a kind in its own right.

Every direction has an "opposite" property, which is always another direction. These occur in matched pairs. The opposite of north is south, just as the opposite of south is north. The opposite of southeast is northwest, the opposite of inside is outside, and so on. When Inform reads a sentence like...

```
Bangkok is south of Nakhon Sawan.
```

...it assumes that the opposite map connection is probably also valid, so that

```
Nakhon Sawan is north of Bangkok.
```

The chapter began with the twelve directions built into Inform:

```
north, northeast, east, southeast, south, southwest, west, northwest,
up, down, inside, outside
```

But the built-in set is not always appropriate. Sometimes this is too many; if we wanted to write about a Flatland, for instance, then up and down ought to go. But in practice it is better not to abolish them as directions but instead to forbid travelling in them. (See the Recipe Book for examples.)

But away from our familiar Earth, the usual frame of reference loses its meaning. Terry Pratchett's "Discworld" comedies, set on a rotating disc, use the directions turnwise, widdershins, hubwards and rimwards. On board a Zeppelin airship, which constantly changes its course, the cockpit has no fixed compass bearing from the passenger cabin: it is not very naturally "north". In zero gravity, there is no up or down. Mars does not have a magnetic core, so a compass doesn't work there.

New directions must always be created in opposing pairs, and each **must** be declared with a clear simple sentence of the form "X is a direction." For instance:

```
Turnwise is a direction. The opposite of turnwise is widdershins.
Widdershins is a direction. The opposite of widdershins is turnwise.
Hubwards is a direction. The opposite of hubwards is rimwards.
Rimwards is a direction. The opposite of rimwards is hubwards.
```

It is then possible to write, say, that:

```
Ankh-Morpork is hubwards of Lancre and turnwise from Borogravia.
```

Of course the Map page of the Index for the project normally draws a map based on compass bearings, so it will get a little befuddled by this. But the map drawn in the Index can be given hints to improve its legibility. More on this later, but for now note that

```
Index map with turnwise mapped as east.
```

maps turnwise directions as if they were east, that is, pointing rightwards on the page. (This has no effect on the story file produced; it does not mean turnwise is simply a new name for east; it affects only the look of the Index map, which is only a convenience for the author in any case.)

At one time, directions had to have shortish names (up to three words only), but that's no longer true:

```
Just the tiniest smidge off magnetic north is a direction. The opposite
of
just the tiniest smidge off magnetic north is just the tiniest smidge
off magnetic south.
Just the tiniest smidge off magnetic south is a direction. The opposite
of
just the tiniest smidge off magnetic south is just the tiniest smidge
off magnetic north.
```

Start of Chapter 3: Things

Back to §3.25. The location of something

Onward to Chapter 4: Kinds: §4.1. New kinds

Example 40: ★★ **Prisoner's Dilemma** A button that causes a previously non-existent exit to come into being.

Example 41: ★★ **The World of Charles S. Roberts** Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.

Example 42: ★★★ **Fore**
Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

Examples from Chapter 3: Things

Start of this chapter
Chapter 4: Kinds
Indexes of the examples

★ **ExampleBic** *RB*
Testing to make sure that all objects have been given descriptions.

It may occasionally be useful to check whether all objects in our game have a given property. Here we have a "not for release" section that will run at the start of the game and alert us to any objects lacking description:

```
"Bic"

Section 1 - Testing descriptions - Not for release

When play begins (this is the run property checks at the
start of play rule):
    repeat with item running through things:
        if description of the item is "":
            say "[item] has no description."

Section 2 - Story
```

```
The Staff Break Room is a room.

The player carries an orange, a Bic pen, and a napkin.
The description of the orange is "It's a small hard pinch-
skinned thing from the lunch room, probably with lots of
pips and no juice."

The description of the napkin is "Slightly crumpled."
```

Example **Verbosity 1** *RB*

Making rooms give brief room descriptions when revisited.

By default, the description of a room is printed every time the player enters a room.

On a device with very limited screen space, however, we might wish to supplant that behavior with "brief" descriptions. In Brief mode, Inform prints room descriptions only when the player enters that room for the first time. Afterwards, the text is skipped, for brevity, though the player can see it again at any time by typing LOOK.

As we saw in the previous chapter, we can set "use options" to control certain aspects of the player's experience. One of the use options is the option to

```
Use brief room descriptions.
```

which changes the defaults so that the description of a room is printed only the first time the player enters.

```
"Verbosity"

Use brief room descriptions.
```

```
The Wilkie Memorial Research Wing is a room. "The
research wing was built onto the science building in
1967, when the college's finances were good but its
aesthetic standards at a local minimum. A dull brown
corridor recedes both north and south; drab olive doors
open onto the laboratories of individual faculty
members. The twitchy fluorescent lighting makes the
whole thing flicker, as though it might wink out of
existence at any moment.

The Men's Restroom is immediately west of this point."

The Men's Restroom is west of the Research Wing.
"Well, yes, you really shouldn't be in here. But the
nearest women's room is on the other side of the
building, and at this hour you have the labs mostly to
yourself. All the same, you try not to read any of the
things scrawled over the urinals which might have been
intended in confidence."

Test me with "west / east".
```

If we type "test me" during play, these commands will be carried out automatically, and we can see that when we return to the Research Wing, the description is not given a second time.

Some notes: the player can also turn full-length descriptions on or off with the commands "verbose" and "brief", or set a minimal-description setting with the command "superbrief". This power still belongs to the player even if we have set the use option to show brief room descriptions by default.

Moreover, we can ourselves check what the state of the descriptions is, with

```
if set to sometimes abbreviated room descriptions: ...
if set to unabbreviated room descriptions: ...
if set to abbreviated room descriptions: ...
```

Finally, it is possible to exercise more precise control over what the player sees on his first and subsequent visits to a room; see the next example for details.

Example **Slightly Wrong** *RB*

A room whose description changes slightly after our first visit there.

A fairly common effect in interactive fiction is a room which is described differently on the first visit than on subsequent visits. We can produce this effect as follows:

```
"Slightly Wrong"

Awning is a room. "A tan awning is stretched on tent poles over the dig-site, providing a little shade to the workers here; you are at the bottom of a square twenty feet on a side, marked out with pegs and lines of string. Uncovered in the south face of this square is an awkward opening into the earth."

Slightly Wrong Chamber is south of the Awning. "[if unvisited]When you first step into the room, you are bothered by the sense that something is not quite right: perhaps the lighting, perhaps the angle of the walls. [end if]A mural on the far wall depicts a woman with a staff, tipped with a pine-cone. She appears to be watching you."

Test me with "look / s / look".
```

Note the "[if unvisited]..." in the description of the Slightly Wrong Chamber. A room is considered to be "unvisited" until after the player has seen its description for the first time.

The bracketed text creates a special rule for printing; we will learn more about these in the sections on text with variations and text with

substitutions.

Some further fine print: we might write our condition as "if unvisited", "if the location is unvisited", or "if the Chamber is unvisited" -- all of these constructions would be acceptable, but in the absence of more specifics, the condition is understood to apply to the object whose description it is.

Example **Port Royal 1** *RB*

A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

"1691"

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Unless we arrange otherwise, this will be the first room in the game because it is the first we have defined.

For subsequent rooms, we do not have to say explicitly that they are rooms, as long as they are connected to a room on the map. For instance, this will automatically make Thames Street End a room:

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

> Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east.

> Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in Port Royal from its earliest days, it is nearly the largest building in the town."

If we have some concern that the room name will be confused with an existing name, we can be more explicit about it using "called":

> East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

> The Private Alley is south of Thames Street at the Wherry Bridge. "You're just outside the tavern the Feathers. To the north, under a pretty little archway, is the active mayhem of Thames Street, but the alley narrows down to a dead end a little distance to the south."

And now we get "inside", which generates a space treated as its own area on the map.

> The Feathers is inside from the Private Alley. "Newly built with brick, replacing the older Feathers tavern that used to stand here. It sells wines in quantity, as well as serving them directly, and the goods are always of the best quality. There's a room upstairs for those wanting to stay the night." The Feathers Bedroom is above the Feathers.

And if we like we can declare a number of rooms for which we will come back and write the descriptions later. There is no obligation for the

description to occur at the first definition of the room.

```
Lime Street is south of Thames Street End.
```

For efficiency, we can also write multiple sets of connections at once:

```
Queen Street East is east of Queen Street Middle and
south of Private Alley.
```

Clicking Go will translate this description into a sketchy but working simulation of Port Royal, in which we can type movement commands like EAST or SOUTH to explore the streets. Looking at the World tab of the Index, we can also see a schematic map of the simulation as it currently stands. Like the rest of the Index, this is provided entirely for the author's benefit, and is not visible to the player. (Though if we do decide that we want players to have access to a printed map while they play, Inform can help: we will return to the layout of Port Royal in the chapter on Publishing.)

The following Test command allows us to type TEST ME and explore the map we just devised:

```
Test me with "s / e / e / s / in".
```

Example **Up and Up** *RB*

6 Adding a short message as the player approaches a room, before the room description itself appears.

Sometimes when a player moves from one room to another, we want to imply that a considerable amount of time elapses, or that something interesting occurs on the way. In that case, we might want to print more than just the room description itself. Here is how we might define a couple of rooms that are far apart:

```
"Up and Up"
```

```
The Plain of the Skull is below the Endless Tower. The
description of the Plain of the Skull is "A vast and
trackless plain, enlivened only by the bones of those
who have previously tried and failed to cross. Above you
is the Endless Tower, which rises half-way to the
moon."
```

```
The description of the Endless Tower is "From up here
the Plain of the Skull seems only a small bald patch: the
world is round and most of it is covered with trees. Far
off to the southwest is a shimmering surface that might
be water; but there are no signs of cities or civilizations,
only the lizard-skeletons."
```

And now we borrow from the instructions on Actions to create our actual message. "Before..." introduces a rule that occurs when the player tries to do something; in this case, we will make a Before rule for going to the tower.

```
Before going to the Endless Tower:
    say "You climb... and climb... and climb... The sun
sets. The moon rises. The wind begins to blow. You
continue to climb..."
```

```
The player carries a bit of harness. The description of
the harness is "A strip of worked leather and a loop of
metal, scavenged from one of the skeletons on the
plain. Without it, you might think your entire quest was in
vain."
```

```
Test me with "look / up".
```

Example **Starry Void** *RB*

Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

Sometimes we may want a room to be visible from the outside in one location, but treated as a separate location when we are inside. The simplest way to do this is to make the exterior form of the object into a door object, and to describe it differently from different vantage points. (Doors in general are described more fully in the Doors section of the Things chapter.)

```
"Starry Void"

The Center Ring is a room.

The magician's booth is a door. "[if the player is in
Center Ring]A magician's booth stands in the corner,
painted dark blue with glittering gold stars.[otherwise if
the magician's booth is closed]A crack of light indicates
the way back out to the center ring.[otherwise]The door
stands open to the outside.[end if]".
```

Here we've arranged for the booth to be described in the initial room description in different ways depending on where the player is when viewing it. We might like to do the same if the player takes a closer look:

```
Instead of examining the magician's booth in the Center
Ring:
    say "It is dark blue and glittering with gold stars. [if the
booth is open]The door currently stands
open[otherwise]It has been firmly shut[end if]."

Instead of examining the magician's booth in the Starry
Void:
    say "The booth door is [if the magician's booth is
open]wide open[otherwise]shut, admitting only a thin
crack of light[end if]."
```

And now we put it in place:

```
The magician's booth is inside from Center Ring and
outside from Starry Void.
```

...and make sure that the booth-and-door object responds to all the names we have used for it in different places:

```
Understand "door" or "of" or "the" or "light" or "crack" or
"thin crack" as the booth.
```

```
Test me with "examine booth / open door of the booth /
in / examine door / close door / look / examine crack of
light".
```

A final nice touch, if we're so inclined, is to borrow from the Basic Actions chapter and make the player automatically open the booth door before trying to enter:

```
Before going through the closed magician's booth:
    say "(first opening the door of the booth)[command
clarification break]";
    silently try opening the booth.
```

For the contrasting case of a space that is nested inside another place and is not its own room -- say a stall at an open-air market, or a rowboat on a lake -- see the example "Tamed".

ExamplePort Royal 2 *RB*

Another part of Port Royal, with less typical map connections.

```
"1691"
```

```
Thames Street End is a room.
```

If we check out a map of historic Port Royal, we find that Thames Street End bends around the northwest tip of the peninsula and becomes the (very) roughly north/south Fisher's Row. We can't put Fisher's Row south of Thames Street End, though, because Lime Street is already going that way. So instead, let's have a map connection that bends around from west to north:

```
West of Thames Street End is north of Fisher's Row.
```

Now continuing west along Thames Street, or north along Fisher's Row, will bring us around the corner in question. Asymmetric map connections should be used carefully. They're good for representing the layout of the real world, which tends not to be laid out on a convenient square matrix, but if exits are not described clearly they can be disorienting for the player. So let's be sure to make things clear:

```
The description of Fisher's Row is "A waterfront street
that runs south towards Chocolata Hole, where the
small craft are harboured. It also continues north around
the tip of the peninsula from here, turning into the east-
west Thames Street."
```

Meanwhile, suppose Fort James is in a prominent position, raised a bit from its surroundings; maybe the player should be able to go down from there, as well as south, to get to Thames Street End.

```
Thames Street End is down from Fort James. Thames
Street End is south from Fort James.
```

But we don't want the upward direction to work:

```
Up from Thames Street End is nowhere.
```

```
Test me with "n / d / u / w / e / n / s".
```

Example The Unbuttoned Elevator Affair *RB*

A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

This is very simple. The interior of the elevator is a single room, but which is mapped east of both of its termini. The reverse map connection, west from the elevator, can only go to a single room, and that's what determines which floor the elevator is on.

```
"The Unbuttoned Elevator Affair"


UNCLE Headquarters is a room. "The steel nerve-
center of the free world's battle against the
Technological Hierarchy for the Removal of
Undesirables and the Subjugation of Humanity. Being
against technology, we have only a very simple elevator
to the east."


Del Floria's Tailor Shop is a room. "Only trained anti-
THRUSH agents recognise the booth in the east wall as
a secret elevator."


The Secret Elevator is east of UNCLE Headquarters.
The Secret Elevator is east of Del Floria's Tailor Shop.


After going to the Secret Elevator:
    say "The doors automatically close, there is a rush of
motion, and they open again.";
    if UNCLE Headquarters is mapped west of the Secret
Elevator, now Del Floria's Tailor Shop is mapped west
of the Secret Elevator;
    otherwise now UNCLE Headquarters is mapped west
of the Secret Elevator;
    continue the action.
```

Test me with "east / west / east / west".

---

10 ExamplePort Royal 3 *RB*
Division of Port Royal into regions.

"1691"

We should go ahead and do all our room definitions first...

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Most of that stretch is full of people at all hours. Imported goods are moved off of ships and taken to distributors; exported goods are brought to be loaded; and there is one public house and brothel for every ten inhabitants.

Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

Lime Street is south of Thames Street End. West of Thames Street End is north of Fisher's Row. The description of Fisher's Row is "A waterfront street that

runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west Thames Street."

Thames Street End is down from Fort James. Up from Thames Street End is nowhere.

Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east.

Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in Port Royal from its earliest days, it is nearly the largest building in the town."

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Fishmarket is southwest of Thames Street at the Wherry Bridge.

The Private Alley is south of Thames Street at the Wherry Bridge. "You're just outside the tavern the Feathers. To the north, under a pretty little archway, is the active mayhem of Thames Street, but the alley narrows down to a dead end a little distance to the south."

The Feathers is inside from the Private Alley. "Newly built with brick, replacing the older Feathers tavern that used to stand here. It sells wines in quantity, as well as serving them directly, and the goods are always of the best quality. There's a room upstairs for those wanting to stay the night." The Feathers Bedroom is above the Feathers.

Thames Street by the King's House is east of Thames Street at the Wherry Bridge. "The King's House is reserved for the use of the Governor, but he does not live in it, and it is frequently being rented out to some merchant so that the government will at least derive some value from it. It is nearly the least interesting establishment on Thames Street, and the crowd -- which, to the west, is extremely dense -- here thins out a bit."

Thames Street before Fort Carlisle is east of Thames Street by the King's House. "Here Thames Street, formerly a respectable width, narrows to a footpath in order to edge around the front of Fort Carlisle, underneath the mouths of the cannon.

There are no buildings on the harbour side of Thames Street at this point, which means that you have an unusually good view of the ships at dock, water beyond, and the Blue Mountains rising on the other side of the harbour."

South of Thames Street before Fort Carlisle is a room called Fort Carlisle. The description of Fort Carlisle is "Handsomely arrayed with cannons which you could fire at any moment -- though of course there are ships at dock which might be in the way."

Queen Street End is south of Lime Street.

```
Queen Street Middle is east of Queen Street End.

Queen Street East is east of Queen Street Middle and
south of Private Alley.

Queen Street at the Prison is east of Queen Street East.
```

Now, if we like, we can create regions to distinguish the coast from the portions of town that aren't on the water:

```
Inland is a region. Queen Street End, Queen Street
Middle, Queen Street East, Private Alley, Lime Street,
and Queen Street at the Prison are in Inland.

Waterfront is a region. Thames Street before Fort
Carlisle, Thames Street by the King's House, Thames
Street at the Wherry Bridge, Water Lane, Fishmarket,
Fisher's Row, and Thames Street End are in Waterfront.
```

There's no rule that regions must be contiguous, so we could if we like make a region consisting just of the two forts:

```
Military Holdings is a region. Fort Carlisle and Fort
James are in Military Holdings.
```

And we might make the Feathers Tavern part of the Inland area, but within its own subcategory:

```
Tavern is a region. It is in Inland. Feathers and Feathers
Bedroom are in Tavern.
```

Now the index map will be colored to reflect our regions, and later in the game development we would be able to make rules that affect just one region at a time.

11 Example **First Name Basis**

RB

Allowing the player to use different synonyms to refer to something.

Sometimes we create objects that we want the player to be able to call by different names: a television that should also answer to "tv" and "telly", for instance, or a refrigerator the player might also call "fridge". In this case, we can use instructions like

```
Understand "tv" and "telly" as the television.
```

to add extra names to the object we've defined.

```
"First Name Basis"

The Crew Lounge is a room. "Deliberately spartan: the crew feels weight restrictions here first, so there aren't any chairs, just a few thin pads on the ground."

The holographic projector is a device in the Crew Lounge. "The one major source of entertainment is the holographic projector, a top of the line Misthon 9000, on which you view every beam you can get." Understand "holo" or "holograph" or "Misthon" or "9000" as the projector.

The description of the projector is "[if switched on]The projector is now playing a documentary about the early politics of the Mars colony.[otherwise]The air above the projector is disappointingly clear.[end if]".
```

(This description is for local color; we will learn more about devices, and conditions like "if switched on", later in this chapter.)

By default, Inform does not understand the names of an object's kind as

referring to that object, unless the object has no other name of its own. We can change this, if we like, by defining names that should be applied to everything of a given kind:

```
Lewis and Harper are men in the Crew Lounge.
Understand "man" or "guy" or "chap" or "lad" or "male"
as a man. Understand "men" or "chaps" or "lads" or
"guys" or "males" as the plural of a man.


The description of Lewis is "A wiry, excitable engineer
who just signed aboard last week." The description of
Harper is "Harper's a good guy: taciturn when sober,
affectionate when drunk, but rarely annoying in either
state."


Test me with "x holo / x man / lewis / x guy / harper /
turn on projector / x holo projector / get men".
```

Inform's naming abilities go considerably further, in fact: we can also instruct it to understand words only under certain circumstances, or only when they appear with other words. Fuller details may be found in the chapter on Understanding.

Example **Midsummer Day** *RB*

A few sentences laying out a garden together with some things which might be found in it.

```
"Midsummer Day"


East of the Garden is the Gazebo. Above is the
Treehouse. A billiards table is in the Gazebo. On it is a
trophy cup. A starting pistol is in the cup. In the
Treehouse is a container called a cardboard box.
```

Test me with "up / x box / d / e / x table / x cup / x pistol / get cup".

---

13 **ExampleTamed** *RB*

Examples of a container and a supporter that can be entered, as well as nested rooms.

Within a room, we might have containers and supporters that a player can enter. A chair, stool, table, dais, or pedestal would be an enterable supporter (anything we would describe a person as being "on"); a cage, hammock, or booth would be an enterable container (because we would describe the person as being "inside").

When the player is in or on something, he is able to see the rest of the contents of the room, but a note such as "(in the hammock)" or "(on the poster bed)" is added to the room title when he looks around.

Here is an example to show off the possibilities:

```
"Tamed"

The Center Ring is a room. The cage is in the Center
Ring. A lion is an animal in the cage. The cage is
enterable, openable, transparent, and closed.
```

Notice that we made the cage transparent. Strictly speaking it is not made of transparent materials, but we can see into (or out of) a closed cage due to the gaps between the bars, so that from Inform's point of view a cage behaves much like a large sturdy glass box. (If we really wanted to make a distinction between, say, an airtight container and one with perforations, we could do so, but Inform does not model such nuances by default.) If a container is not transparent, we can see into and out of it only when it is open.

Supporters are a bit more straightforward because there is no circumstance in which they separate the player from the rest of the world:

```
The pedestal is in the Center Ring. It is enterable.
```

And in fact we can tell Inform that the player starts on the pedestal with this line:

```
The player is on a pedestal.
```

Now the player will begin there rather than just in the Center Ring.

This last bit is an entirely unnecessary bit of local color, but if we're going to keep getting into and out of the lion's cage, we ought to expect him to take notice:

```
Every turn when the player is in the cage:
    if a random chance of 1 in 2 succeeds, say "The lion
eyes you with obvious discontent.";
    otherwise say "Though the lion does not move, you
are aware that it is watching you closely."
```

Randomness is explained more completely in the chapter on Change, and every turn rules in the chapter on Time.

Finally, we might want a container whose interior is modeled as its own separate room: say, a magician's booth in which volunteers are made to disappear.

```
The magician's booth is a container in Center Ring. "Off
to one side is a magician's booth, used in disappearing
acts. The exterior is covered with painted gilt stars." The
booth is enterable, open, not openable, and fixed in
place.
```

Now we create our other location:

```
Inside from the Center Ring is the Starry Vastness.
```

...which handles the case of the player typing >IN. (We will not assume by default that he wants to get into the cage with the lion, this being obviously perilous.) But we also want to make sure that the player who types >ENTER BOOTH winds up in the same place, so we should add:

Instead of entering the magician's booth: try going inside.

Test me with "get in cage / open cage / get in cage / z / close cage / out / open cage / get on pedestal / get off / look / enter booth / out".

---

14 **Example** **Disenchantment Bay 1** *RB*

A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

To begin with the title:

"Disenchantment Bay"

There are many Disenchantment Bays across the world, named by eighteenth-century ships' captains - one in Antarctica, another in Tasmania, for instance. The most famous is probably the one where Lewis and Clark's expedition broke through to the Pacific. But ours is the one in Alaska, named in 1791 by a Spanish navigator who had hoped it might lead to the fabled Northwest Passage, and all of this history is beside the point since the game is set in the present day.

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the

```
Yakutat Charter Boat Company."
```

We might want to start with the glass case.

```
The Cabin contains a glass case. In the glass case is a
collection of fishing rods.
```

Now Inform will have guessed that the case is a container, but its default idea of a container is something like a bucket: permanently open and not able to be opened and shut. We can change that:

```
The case is closed, transparent, and openable.
```

We get a similar set of guesses if we write

```
The bench is in the cabin. On the bench are some blue
vinyl cushions.
```

Using "some" rather than "a" or "the" tells Inform that the cushions are to be referred to as a plural object in the future. And because of the "on the bench..." phrase, Inform will guess that the bench is a supporter and that it is fixed in place and cannot be moved from room to room. We do have to tell it that the bench can be sat on, though:

```
The bench is enterable.
```

And now a short script, so that if we type TEST ME, we experiment with the case and bench:

```
Test me with "examine case / get rods / open case / get
rods / sit on bench / take cushions / get up"
```

ExampleDisenchantment Bay 2 *RB*

Disenchantment Bay: creating some of the objects in the cabin's description.

If we compile our last version of the cabin, we get a room where the glass case and the bench are listed separately from the room description, even though they have already been mentioned once. We can prevent this by making the already-mentioned things scenery:

```
"Disenchantment Bay"

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.
```

Generally speaking, it is a good idea to recognize the player's attempts to interact with any objects mentioned in the room description, so we should also provide

```
Some navigational instruments, some scratched windows, a sign, a radar display, and some radios are scenery in the cabin.
```

```
Test me with "examine instruments / x windows / x sign
/ x display / x radios".
```

The door and the view will need to be done as well, but they are special cases which we will get to shortly.

As noted, making something scenery also means that the player will be prevented from picking it up and carrying it away. This is sensible, though: if an object can be removed from the room where it first appears, we should be careful about mentioning it in the main room description; otherwise, it will continue to be described as present even when someone has carried it off.

---

16 ExampleReplanting *RB*

Changing the response when the player tries to take something that is scenery.

By default, "TAKE OAK" in the example above will produce the response "That's hardly portable." This is fine under many circumstances, but also a bit generic, so we might want to override it for a specific game.

```
"Replanting"

The Orchard is a room. "Within this quadrille of pear
trees, a single gnarled old oak remains as a memory of
centuries past." The gnarled old oak tree is scenery in
the Orchard.

Instead of taking some scenery: say "You lack the hulk-
like strength."

Test me with "take oak".
```

Here we've used an "instead" rule; we will learn more about these in the

section on actions. This allows us to define our own results for taking an object.

Note: "scenery" is a property of an object (about which we will hear more later). So when we use it in rules, we can talk about "some scenery", "something that is scenery", or even "a scenery thing" -- the last one doesn't sound much like English, but is a more plausible construction with other adjectives.

---

17 **Example** **Disenchantment Bay 3** *RB*

Disenchantment Bay: adding a view of the glacier.

Suppose we wanted to have the glacier visible from the Cabin of our boat, and anywhere else we might add to the game:

```
The view of the Malaspina glacier is a backdrop. It is
everywhere. The description is "The Malaspina glacier
covers much of the nearby slope, and -- beyond it -- an
area as large as Rhode Island."
```

---

18 **Example** **Disenchantment Bay 4** *RB*

Disenchantment Bay: fleshing out the descriptions of things on the boat.

Currently we have provided objects for most of what is on the boat, but it's not very interesting to look at. We might want to give some more description to these things.

```
"Disenchantment Bay"

The Cabin is a room. "The front of the small cabin is
entirely occupied with navigational instruments, a radar
display, and radios for calling back to shore. Along each
side runs a bench with faded blue vinyl cushions, which
```

can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin.

The description of the instruments is "Knowing what they do is the Captain's job."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now."

The description of the radar is "Apparently necessary to avoid the larger icebergs."

```
The description of the radios is "With any luck you will
not need to radio for help, but it is reassuring that these
things are here."
```

The order in which we define these things is fairly open. We could also define an object so:

```
A sign is scenery in the Cabin. The description is "You
can get half-day and full-day sight-seeing tours, and
half-day and full-day fishing trips."
```

Where "the description" is assumed to refer to the thing most recently defined, if no object is specified.

```
The view of the Malaspina glacier is a backdrop. It is
everywhere. The description is "The Malaspina glacier
covers much of the nearby slope, and -- beyond it -- an
area as large as Rhode Island."
```

```
Test me with "examine sign / examine glacier / examine
instruments / examine windows / examine radar /
examine radios / take the cushions / take the glacier".
```

These last two commands show how scenery and backdrops are automatically impossible for the player to take.

ExampleLaura *RB*

Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

Occasionally it is useful to give something a printed name because we want to call it something extremely long-winded; give one thing a name that is the subset of the name of something else; or use words such as "with" or "and" that are likely to confuse Inform into thinking that the object name ends before it actually does.

Often it is enough to preface these ambiguously-titled things with "a thing called..." or "a supporter called..." or the like, as here:

```
South of Spring Rolls is a room called Hot and Sour
Soup.
```

prevents Inform from trying to read "Hot and Sour Soup" as two separate rooms, while

```
The player carries an orange ticket. The player carries a
thing called an orange.
```

creates two objects instead of the one orange ticket that would result if the second sentence were merely "The player carries an orange."

Really long names can be a bit cumbersome. For example:

```
The player carries a thing called an incriminating
photograph of a woman with blonde hair.
```

So we might instead give the photograph a printed name:

```
"Laura"
```

```
The City of Angels is a room. The incriminating
photograph is carried by the player. The printed name of
the incriminating photograph is "incriminating
photograph of a woman with blonde hair".
```

Now we've gotten around any awkwardness with printing the name -- but we also need to understand when the player refers to the photograph. When we define the names of objects under normal circumstances, Inform takes care of this automatically, but if we have especially set the printed name, we must also specially define the appropriate terms for the player to use. For this we need "understand", which will be explained in much more depth in a later chapter:

```
Understand "woman" or "with" or "blonde" or "hair" or
"of" or "a" as the incriminating photograph.
```

```
Test one with "x photograph / x incriminating
photograph of a woman with blonde hair / x hair / x
blonde / x woman with blonde hair / x incriminating
photograph of a woman".
```

That's probably as far as we really need to go, and if you are satisfied with this behavior, there is no need to read on.

One possible objection to this solution is that Inform will accept some nonsensical formulations as applying to the photograph: for instance, it will allow >EXAMINE PHOTOGRAPH OF, >X BLONDE PHOTOGRAPH WOMAN INCRIMINATING, or even >X OF ...though in the case there were two items with "of" names, the game would disambiguate with a question such as "Which do you mean, the incriminating photograph of a woman with blonde hair or the essence of wormwood?"

Traditionally, Inform has tended to be fairly flexible about word order, preferring to err in the direction of leniency. On the other hand, there are times when we need more exacting rules in order to distinguish otherwise similar cases.

Two features allow us to specify more exactly if we so desire. The first is that, if we specify a whole phrase as the name of something, all the words in that phrase are required, in the order given. Thus "Understand "blonde hair" as the photograph" would require that both "blonde" and "hair" be present, and would not recognize >X BLONDE, >X HAIR BLONDE, or >X HAIR.

Second, we can create tokens, such as "Understand "blonde hair" or "hair" as "[hair]", and then use these tokens in match phrases. This saves a good deal of time when we want to specify a number of different but fussy alternatives. So, for instance, here is a drawing that would not respond to >X OF, or >X BROWN EYES, but would respond to >X DRAWING OF MAN WITH BROWN EYES, >X MAN WITH BROWN EYES, and so on:

```
The drawing is carried by the player. The printed name
of the drawing is "drawing of a man with brown eyes".


Understand "eyes" or "brown eyes" as "[brown eyes]".
Understand "man" or "man with [brown eyes]" or
"brown-eyed man" as "[man]". Understand "[man]" or
"drawing of [man]" or "drawing of a [man]" as the
```

```
drawing.

Test me with "test one / test two".

Test two with "x drawing / x man / x of / x drawing of
man / x drawing of a man / x drawing of a man with
brown eyes / x drawing of a brown-eyed man / x brown
eyes".
```

Further refinements are possible: the "privately-named" attribute tells Inform not to try to understand the source name of an object at all, so if we write

```
The purple rabbit is a privately-named thing.
```

...the player will not be able to refer to it as "purple" or "rabbit" or "purple rabbit".

There are also ways to make names to refer to entire kinds of objects (so "dude" will refer to any man in the game); to specify names that only refer to objects in the plural (so GET PICTURES will pick up several pictures together); to reflect an object's properties (so "red apple" works only as long as the apple is in fact red); or even to refer to the object's relationships to other objects (so "bottle of wine" works only when wine is indeed in the bottle). All these refinements are discussed in the chapter on Understanding.

ExampleDisenchantment Bay 5 *RB*

Disenchantment Bay: adding the door and the deck to our charter boat.

We mentioned that there is a door out to the deck in our example. The following two sentences will create both the door and the other room:

```
The cabin door is south of the Cabin and north of the
Deck. It is a door and scenery.
```

Now Inform has constructed a generic room called "Deck" to the south. It has neither a description nor any contents yet, but we could fix that in time. It does have a view of the glacier, though, since we defined the glacier view to be everywhere.

---

21. **ExampleEscape** *RB*
Window that can be climbed through or looked through.

Suppose we want to offer the player a window he can climb through, instead of a boring ordinary door. Our window will be like a door in that it connects two rooms, appears in both places, and impedes movement when it is shut. But we also want to add that we can look through it and see what lies on the other side; and we further want to understand "climb through window" or "jump through window" as attempts to pass through it.

We'll start by defining a couple of rooms and making the window a door between them.

```
"Escape"

Your Bedroom is a room. The bedroom window is a
door. It is west of Your Bedroom and east of the Grassy
Slope.
```

Now we have a "bedroom window" object which can be entered. Now, to catch the case where the player types "LOOK THROUGH WINDOW":

```
Instead of searching the window:
    say "Through the window, you make out [the other
side of the window]."
```

The other side of a door is always defined to be the room that we are not currently in when doing the check. When we are in the bedrooom, the other side will be the grassy slope, and vice versa. "Searching" is the action that occurs when the player attempts to LOOK THROUGH something. (To review what grammar gives rise to what actions, we can always consult the Actions portion of the Index.)

Next we want to cover the case where we climb through the window:

```
Instead of climbing the window:
    try entering the window.
```

And because "climb window" is understood but "climb THROUGH window" is not, we will have to borrow from the chapter on Understanding to add some new vocabulary to the game (and we'll add Jump too, while we're at it):

```
Understand "climb through [something]" as climbing.
Understand "jump through [something]" as climbing.
```

Now the final piece: Inform will already keep the player from going through a closed window, but it will say "You can't, since the bedroom window is in the way." This is probably not ideal, so we can replace the instruction thus:

```
Instead of going through the closed window:
    say "The window is shut: you'd break the glass."
```

```
Test me with "look through window / climb through
window / open window / climb through window / look
through window / close window / e / open window / e".
```

ExampleGaribaldi 1 *RB*

Providing a security readout device by which the player can check on the status of all doors in the game.

Suppose we would like to allow the player to view the status of all the doors functioning in the game; and we want to identify those doors by mentioning which two rooms they connect. The following uses some techniques that will be covered in later chapters, but the basic idea may be obvious:

```
"Garibaldi"


The security readout is a device. The description of the
readout is "The screen is blank."


Instead of examining the switched on security readout:
    say "The screen reads: [fixed letter spacing]";
    say line break;
    repeat with item running through doors:
        say line break;
        say " [item] ([front side of the item]/[back side of the
item]): [if the item is
locked]LOCKED[otherwise]UNLOCKED[end if]";
    say variable letter spacing;
    say paragraph break.
```

It is more or less arbitrary which room winds up as the "front side" and which as the "back", but in this case it hardly matters.

```
The player carries the security readout.


The Docking Bay is a room. The inner airlock is a door.
It is north of the Docking Bay and south of the Zocalo.
The inner airlock is lockable and unlocked. The outer
airlock is lockable and locked. It is a door. It is south of
the Docking Bay and north of Space.


The quarantine seal is a door. It is west of the Zocalo
and east of Medlab. Quarantine seal is locked.


The security pass unlocks the inner airlock. The player
carries the security pass.


Test me with "x readout / turn on readout / x readout /
```

lock inner airlock with security pass / x readout".

---

23 **Example** **Disenchantment Bay 6** *RB*

Disenchantment Bay: locking up the charter boat's fishing rods.

It stands to reason that the captain wouldn't let just anyone meddle with his fishing equipment; maybe he keeps that case locked. We could replace the case description with this one, instead:

> The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

Now there's a silver key that will unlock it -- though since we haven't said where the key is, the player will never be able to find it in the game. (If we look at the World index, we find "small silver key" right at the bottom, not inside any of the rooms. That is as good as not existing at all -- though we usually use the term "out of play" -- but as we will later see, it is possible to have things initially out of play but brought into existence later on.)

---

24 **Example** **Neighborhood Watch** *RB*

A locked door that can be locked or unlocked without a key from one side, but not from the other.

Suppose we want a locked door that can be opened with a key, but is also openable by hand without a key from one side only. We start by defining an ordinary lockable door and the key that controls it:

> "Neighborhood Watch"
>
> The shabby door is a door. It is outside from the Studio

```
Apartment and inside from the Rickety Stairwell. The
shabby door is locked.


The brass key is carried by the player. It unlocks the
shabby door.
```

The next part is going to require that we modify the normal operation of the "lock" command. "Lock" ordinarily requires that the player supply two objects: a thing he wants to unlock, and the key he wants to use on it. The full command is LOCK DOOR WITH THE KEY, and Inform will not accept simply LOCK DOOR as locking.

Therefore, we're going to need to create our own new variant on the lock verb (and the unlock verb, while we're at it). The full procedure for this is laid out in the chapters on Action and Understanding, but here is an example:

```
Understand "lock [something]" as locking keylessly.
Locking keylessly is an action applying to one thing.
```

Here we've created a new action -- locking something without a key -- and we've told Inform to understand LOCK DOOR as this action, rather than an incomplete command to LOCK DOOR WITH SOMETHING.

Now we add some instructions so that the game will not let us use this keyless unlocking command unless we're in the right place or are properly equipped:

```
Check locking keylessly:
    if the noun is not a door, say "[The noun] is not
something you can lock." instead;
    if the noun is locked, say "[The noun] is already
locked." instead;
    if the player carries the brass key and the player is in
the Stairwell, try locking the noun with the brass key
instead;
    if the player is in the Stairwell, say "You can't lock the
door from this side without the key." instead.
```

This check rule is performed before the keyless locking action succeeds. The first thing we do is try to use the key if the player is outside and has the key: this way, LOCK DOOR will turn automatically into LOCK DOOR WITH THE KEY, under circumstances where that is both possible and

necessary.

The second thing is to check whether the player is outside but keyless, and, if so stop the action from being performed successfully. Here we print a failure message followed by the word "instead", which tells Inform that we've substituted some other outcome for the usual performance of the action.

Now we're reasonably sure that the player is only locking keylessly in the case that he is inside the Studio. (We might have to do a more thorough check for this if there were more than two rooms, but as it is, the player can only be in the Stairwell or in the Studio, so if we have ruled out the Stairwell, we are safe.) So now we want to add what happens when locking-without-a-key command succeeds:

```
Carry out locking keylessly:
    now the noun is locked.
```

That's it. We've just told Inform to make the door be locked. "Now..." syntax will be explained more thoroughly in the chapter on change. But we still haven't described to the player what just happened, so let's provide a description of that, too:

```
Report locking keylessly:
    say "You flip over the deadbolt to lock [the noun]."
```

And now we have to do a similar set of things for unlocking:

```
Understand "unlock [something]" as unlocking keylessly.
Unlocking keylessly is an action applying to one thing.


Check unlocking keylessly:
    if the noun is not a door, say "[The noun] is not
something you can lock." instead;
    if the noun is unlocked, say "[The noun] is already
unlocked." instead;
    if the player carries the brass key and the player is in
the Stairwell, try unlocking the noun with the brass key
instead;
    if the player is in the Stairwell, say "You can't unlock
the door from this side without the key." instead.
```

```
Carry out unlocking keylessly:
    now the noun is unlocked.

Report unlocking keylessly:
    say "You flip over the deadbolt to unlock [the noun]."

Test me with "unlock door / drop key / open door / out /
close door / lock door / open door / in / get key / out /
close door / lock door / unlock door".
```

Some (but not all) of this work is done for you if you like by the Locksmith extension. If you prefer, you can include that extension, then follow the documentation in order to implement the remainder of the scenario. Locksmith takes care of implementing the additional locking and unlocking actions, and provides some other conveniences.

Example **Disenchantment Bay 7** *RB*

Disenchantment Bay: making the radar and instruments switch on and off.

If we would like the player to be able to turn instrumentation on and off, we could add a line to this effect:

```
The radar, the instruments, and the radios are devices.
```

And since the captain is probably not navigating blind, we might also want to say

```
The radar and the instruments are switched on.
```

Suppose we want to have a room with a light switch. Turning the switch off makes the room go dark; turning it on restores the light. This kind of switch is an obvious candidate as a device.

```
"Down Below"

Terrifying Basement is a room. The light switch is a
switched on device in the Terrifying Basement. It is fixed
in place.
```

Here we define our light switch, and we also make it start out as "switched on". The Terrifying Basement will also start out lit (as all rooms do, by default, unless we specifically say that they are dark). We further say that it is fixed in place to avoid the ludicrous possibility of the player picking it up and carrying it away.

Next we add some instructions to control how turning the light switch on and off affects the room light. These borrow from later chapters on actions, but the gist may be obvious anyway:

```
Carry out switching off the light switch: now the
Terrifying Basement is dark.

Carry out switching on the light switch: now the
Terrifying Basement is lighted.
```

Inform already has the idea of light and darkness built in; we will see more about this later, and the Phrasebook (in the Index tab) also contains a list of all the adjectives (lighted, dark, etc) which are important to use here.

Speaking of the Index, the Actions tab contains a list of all the grammar that can be used to activate a given command: for instance, the switching action responds to "switch [something]" or "turn on [something]". In this case, we may want to give the player an extra option or two. It would be pretty natural for a player to try >FLIP SWITCH, so let's add that in:

```
Understand "flip [something switched off]" as switching
on. Understand "flip [something switched on]" as
switching off. Understand "flip [something]" as switching
on.
```

The nuances of this will be explored in the chapter on Understanding. What is useful to know here is that we have taught Inform to understand that >FLIP LIGHT SWITCH means to turn it on when the switch is already off; if the switch is already on, FLIP SWITCH means to turn the switch off. Depending on the kind of device we are modeling (button? lever? dial?), we might want to write similar lines for commands such as PUSH, PRESS, PULL, TURN, and so on.

Finally, we need to deal with a special case. In general, the player cannot interact with other things in a dark room because he can't see them, but if we adhered strictly to this it would be impossible for him to find the light switch to turn it back on. So we need something from the chapter on Activities to change this:

```
After deciding the scope of the player when the location
is the Terrifying Basement:
    place the light switch in scope.


Upstairs is above the Terrifying Basement.


Test me with "turn off light / look / flip light switch".
```

---

27. ExamplePeugeot *RB*

A journey from one room to another that requires the player to be on a vehicle.

Let's say that our protagonist is about to flee . Obviously, he can't make the journey on foot; he needs transportation.

```
"Peugeot"

Include Rideable Vehicles by Graham Nelson.

The Lot is a room. The ten-speed bike is a rideable
vehicle in the Lot.
```

We make the ten-speed bike a rideable vehicle because we want to say that the player is on it rather than in it. Then our other room:

```
Cambridge is east of the Lot.
```

And now we borrow from the Actions chapter to prevent travel without the proper equipment:

```
Instead of going to Cambridge when the player is not on
the ten-speed bike:
    say "It's a long journey to Cambridge: you'll never
make it on foot."

After going to Cambridge:
    say "You begin pedalling determinedly.";
    continue the action.

Test me with "e / get on ten-speed bike / e".
```

ExampleDisenchantment Bay 8 *RB*

Disenchantment Bay: a pushable chest of ice for the boat.

We probably do not need a vehicle to ride around our boat, but there might be a heavy ice chest that can only be pushed from room to room:

```
The ice chest is a closed openable container in the
Deck. "A very heavy ice chest sits on the ground." It is
fixed in place and pushable between rooms. A quantity
of ice is in the chest. The description is "Ready and
waiting just in case there's any fish needing to be kept
cool."
```

This anticipates a later chapter, but it would probably be a good idea to hint to the player, if he tries to take the ice chest, that there is another way to move it:

```
Instead of taking the chest: say "It's too heavy to lift, but
you might be able to push it, and just inch it over the
frame of the door."
```

Otherwise, attempts to pick it up will just reply with "That's fixed in place."

---

29. ExampleHover *RB*

Letting the player see a modified room description when he's viewing the place from inside a vehicle.

Suppose we want the player to see a modified room description when he's viewing the place from inside a vehicle. There are several conceivable ways of doing this; the example here shows a rather advanced way, but is very flexible and will let us write all sorts of special cases.

```
"Hover"

Use full-length room descriptions.

Emerald City is a room. "All the buildings are spires and
none of them have doors." The Vast Desert is west of
Emerald City. "[if the player is in a vehicle]Outside,
```

```
a[otherwise]A[end if] trackless waste stretches as far as
the eye can see in every direction."
```

```
The hover-bubble is a vehicle in the Emerald City. "Your
hover-bubble awaits." The description is "The hover-
bubble is a clear globe-shaped vehicle capable of
transporting you anywhere you could walk, but faster."
Understand "bubble" as the hover-bubble. The hover-
bubble contains a chocolate wrapper and a parking
ticket.
```

Here's the tricky part, which relies on material from the chapters on Activities and Rulebooks:

```
The container interior rule is listed before the room
description body text rule in the carry out looking rules.
```

```
This is the container interior rule:
    if the actor is the player and the player is in an
enterable thing (called current cage), carry out the
describing the interior activity with the current cage.
```

```
Describing the interior of something is an activity.
```

Now we've done that, we can write a "rule for describing the interior" of something, which will print whatever we like:

```
Rule for describing the interior of the hover-bubble:
    say "The hover-bubble is transparent, but tints
everything outside very faintly lavender."
```

In fact, as a special refinement, we could even say:

```
Rule for describing the interior of the hover-bubble when
the hover-bubble contains more than one thing:
    say "The hover-bubble is transparent, but tints
everything outside very faintly lavender. Beside you you
```

can see [a list of other things in the hover-bubble]."

Definition: a thing is other if it is not the player.

Rule for listing nondescript items of the hover-bubble when the player is in the hover-bubble: do nothing.

Test me with "get in bubble / look / west / take all / look / get out / east".

And now anything that's beside us in the vehicle will be described during that first paragraph, rather than later on.

---

30 **ExampleDisenchantment Bay 9** *RB*

Disenchantment Bay: enter the charter boat's Captain.

Now finally we can put a Captain in the boat:

The Captain is a man in the Cabin. "The captain sits at the wheel, steering the boat and occasionally checking the radar readout."

---

31 **ExampleBelfry** *RB*

You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

"Belfry"

```
The Belfry is a room. A bat is in the Belfry. The bell is in
the Belfry. Some woodworm are in the Belfry. A man
called William Snelson is in the Belfry. A woman called
the sexton's wife is in the Belfry. A man called a
bellringer is in the Belfry.


In the Belfry is a man called the vicar. The indefinite
article of the vicar is "your local".


Test me with "look".
```

---

32 **ExampleGopher-wood** *RB*

Changing the name of a character in the middle of play, removing the article.

Suppose that we want a character who starts out with a general epithet ("the bearded man") but is later introduced to the player properly ("Japheth"). In that case, we want to be able to tell Inform to stop using an article once the character has been given a proper name. We can do this like so:

```
"Gopher-wood"


The Ark is a room. A bearded man is in the Ark.


Instead of examining the bearded man for the first time:
    now the printed name of the bearded man is
"Japheth";
    now the bearded man is proper-named;
    say "You peer at him a bit more closely and realize
that it's Japheth."
```

Finally, we need to tell Inform to understand the man's name, but only when he's been introduced. For this purpose, we borrow from the chapter on Understanding:

```
Understand "Japheth" as the bearded man when the
bearded man is proper-named.


Test me with "x japheth / x man / look / x japheth".
```

---

33 **Example Disenchantment Bay 10** *RB*

Disenchantment Bay: things for the player and the characters to wear and carry.

At this point we can dress both the Captain and the player with some appropriate props:

```
The captain wears a baseball cap. The description of
the cap is "It says, THE WORST DAY FISHING IS
BETTER THAN THE BEST DAY WORKING."


The player is carrying a backpack and a bottle of water.
The player is wearing a pair of sunglasses. The
description of the sunglasses is "The light off the water
and the ice does get pretty bright sometimes."
```

(At present the backpack can't be worn, but see the next version.)

---

34 **Example Disenchantment Bay 11** *RB*

Disenchantment Bay: making a holdall of the backpack.

If we wanted, we could make the player's backpack infinitely capacious,

so:

```
The backpack is a player's holdall.
```

...And now whenever the player character is unable to hold everything, he will automatically stow some of his possessions therein.

This is only useful if the player doesn't have infinite carrying capacity himself, so perhaps we also need

```
The carrying capacity of the player is 3.
```

Perhaps mercifully, items which are worn are not counted against the player's carrying capacity. We might want to let him take advantage of that, too:

```
The backpack is wearable.
```

This capacity system makes a compromise between the realistic and the absurd: on the one hand, it acknowledges that people can't carry an infinite number of items in their hands, while at the same time providing a sack that can.

Many games will have no use for object-juggling of this kind at all; others will want to be much more rigorous about questions of capacity and volume. Fortunately, it is easy to leave the whole business out by assigning no carrying capacity to anything.

ExampleFallout Enclosure *RB*

Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

It may not be immediately obvious why we might want to create new intermediate categories of the kinds hierarchy. But there may be times, for instance, where we would like to make an action that applies in the same way to both containers and supporters, but to nothing else in the game. To avoid creating two nearly-identical rules, we would instead roll the two categories together into one, on the principle that duplicating

source text is usually a sign of bad design.

So for instance let's say the player is able to zap objects to make them go away, but any contents -- things inside a container or on top of a supporter -- should always be left as residue. Here's one way we might do this:

```
"Fallout Enclosure"


Section 1 - Procedure


An enclosure is a kind of thing. A container is a kind of
enclosure. A supporter is a kind of enclosure.


Understand "zap [something]" as zapping. Zapping is an
action applying to one thing. The Zapping action has a
list of things called the remnants.


Carry out zapping an enclosure:
    if the noun holds something:
        now the remnants is the list of things held by the
noun;
        repeat with N running through the remnants:
            move N to the holder of the noun.


Carry out zapping:
    now the noun is nowhere.


Report zapping:
    say "You zap [the noun], destroying [them][if the
remnants is not empty] and leaving [the remnants with
indefinite articles] behind[end if]."
```

Section 2 - Scenario

SuperDuperMart is a room. SuperDuperMart contains some shelves and a cash register.

The shelves support a bottle of Buffout and a container of Jet.

The cash register contains some prewar money, a coin purse, and a bottle cap. The coin purse contains a prewar nickel. It is closed.

The cash register is closed and locked.

Test me with "zap shelves / zap buffout / zap register / zap purse".

ExampleBrown *RB*

36

A red sticky label which can be attached to anything in the game, or removed again.

"Brown"

The Shipping Room is a room. The red sticky label is a thing carried by the player. The description of the red sticky label is "It reads: AIRMAIL[if the label is part of something (called the parent)]. It is stuck to [the parent][end if]."

```
A black crate is in Shipping. The description is "A boring
black crate." The brown crate is a thing in Shipping. The
description is "An ordinary brown crate."
```

```
After examining something when the label is part of the
noun:
    say "A bright red sticky label is attached to [the
noun]!"
```

Here is the essential point: whenever we ATTACH LABEL TO something, it becomes part of that object.

```
Instead of tying the red sticky label to something:
    now the red sticky label is part of the second noun;
    say "You stick [the label] to [the second noun]."
```

And of course the label cannot be stuck to itself or to more than one thing at a time.

```
Before tying the label to something when the label is
part of something:
    if the label is part of the second noun:
        say "[The label] is already stuck to [the second
noun]." instead;
    otherwise:
        say "(first freeing the label)[line break]";
        silently try taking the label;
        if the label is part of something, stop the action.
```

```
Instead of tying the red sticky label to the label:
    say "That would ruin the label entirely."
```

```
Instead of taking the label when the label is part of
something:
    now the player carries the label;
    say "You peel the label off again."
```

Much of the rest is just tidying to make sure that the player's commands are redirected into the right syntax.

```
Instead of tying something to the label:
    try tying the label to the noun.

Instead of putting the label on something:
    try tying the label to the second noun.

Instead of inserting the label into something:
    try tying the label to the second noun.

Understand the commands "stick" or "apply" as "tie".
```

We could have created a new "sticking" action, but to keep the example short we will use the built-in "tying" action instead, and respond to the command "stick" just as if it were "tie".

```
Understand "peel [something]" or "peel off [something]"
as taking.

Test me with "i / put label on the black crate / look / x
black / x label / get the label / apply label to brown crate
/ look / x brown / peel off label / stick label to label".
```

Example **Disenchantment Bay 12** *RB*

A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

"Disenchantment Bay"

Include Locksmith by Emily Short.

Use scoring.

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions[if the compartment is closed], which can be lifted to reveal the storage space underneath[otherwise], one of which is currently lifted to allow access to the storage compartment within[end if]. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. Understand "rod" as the collection. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

A storage compartment is an openable closed container. It is part of the bench. Instead of opening the bench, try opening the storage compartment. Instead of

closing the bench, try closing the storage compartment. Instead of pushing or pulling or turning the cushions, try opening the storage compartment. Understand "space" as the storage compartment.

Some nets and a Coke are in the compartment. Understand "net" as the nets. The description of the nets is "They must have something to do with fish as well. Really, you're just here for the sights." The nets are a container.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin. The radar, the instruments, and the radios are devices. The radar and the instruments are switched on.

A screen is part of the radar. The description of the screen is "[if the radar is switched on]Phantom lights move across the screen.[otherwise]The screen is dark.[end if]". Instead of doing something other than examining to the screen, say "It's not good for much but looking at."

The Captain is a man in the Cabin. "The captain sits at the wheel, steering the boat and occasionally checking the radar readout." The captain wears a baseball cap. The description of the cap is "It says, THE WORST DAY FISHING IS BETTER THAN THE BEST DAY WORKING." The captain carries the silver key. The description of the captain is "[The captain] is wearing [a list of things worn by the captain][if the captain carries something] and carrying [a list of things carried by the captain][end if]."

The description of the instruments is "Knowing what they do is the Captain's job." Instead of doing something other than examining to the instruments in the presence of the Captain: say "The Captain glares at you. Clearly

you are not welcome to do that."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now." Understand "window" as the windows.

The description of the radar is "Apparently necessary to avoid the larger icebergs."

The description of the radios is "With any luck you will not need to radio for help, but it is reassuring that these things are here."

A sign is scenery in the Cabin. The description is "You can get half-day and full-day sight-seeing tours, and half-day and full-day fishing trips."

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island." Understand "view of the surrounding bay" or "surrounding bay" as the view.

The cabin door is south of the Cabin and north of the Deck. It is a door and scenery. The description of the Deck is "The whole back half of the boat is open, allowing you to view the surroundings without intervening windows -- if you can stand the cold."

The ice chest is a closed openable container in the Deck. "A very heavy ice chest sits on the ground." It is fixed in place and pushable between rooms.

A quantity of ice is in the Deck. "All around the boat bob chunks of glacier ice." Understand "glacier ice" as the quantity. The description is "Curiously cooled into funny-shaped chunks." The printed name of the quantity is "glacier ice".

Instead of taking the quantity of ice when the player is not carrying the nets:
    if the quantity of ice is handled, continue the action;
    say "You are having a hard time fishing out the ice with your bare hands."

Instead of taking the quantity of ice when the player is carrying the nets:
    if the quantity of ice is handled or the quantity of ice is in the nets, continue the action;
    now the quantity of ice is in the nets;
    say "You scoop up the ice with the net."

Instead of taking the chest: say "It's too heavy to lift, but you might be able to push it, and just inch it over the frame of the door."

The player is carrying a backpack. The player is wearing a pair of sunglasses. The description of the sunglasses is "The light off the water and the ice does get pretty bright sometimes."

The backpack is a player's holdall. The carrying capacity of the player is 3. The backpack is wearable.

Instead of asking the Captain for the key:
    say "'Sure, you can -- well, get me a drink first, would you?'"

Instead of asking the Captain for the key when the Captain is carrying a cold Coke and the Captain is carrying the key:
    move the key to the player;
    say "'Here, knock yourself out.'"

Instead of asking the Captain for the key when the Captain is not carrying the key: say "'I already gave it to you. You didn't lose it, did you?'"

Heat is a kind of value. The heats are cold, cool, room temperature, and warm.

A beverage is a kind of thing. A beverage can be open or closed. A beverage can be openable or unopenable. A beverage is always edible and openable. A beverage has a heat. A beverage is usually warm. The Coke is a beverage. The beer is a beverage. The beer is in the backpack.

Instead of giving or showing a beer to the Captain:
    say "'I don't drink on the job, thanks,' he says. 'You can help yourself if you want it, though.'"

Instead of giving or showing a Coke to the Captain:
    say "'It needs chilling,' the Captain remarks, disgruntled."

Instead of giving or showing a cold Coke to the Captain:
    move the Coke to the Captain;
    increase the score by 2;
    say "'Ah, thank you,' he says. How he can drink an iced soda on a day like today is an open question, but Alaskans are special."

```
Every turn when the quantity of ice is in the ice chest:
    repeat with item running through beverages in the ice
chest:
        let the current heat be the heat of the item;
        if the current heat is not cold, now the heat of the
item is the heat before the current heat.



Before printing the name of a beverage (called the
drink):
    say "[heat of the drink] ".



Understand the heat property as describing a beverage.



The maximum score is 5.



After taking the fishing rods:
    end the story finally;
    increase the score by 3;
    say "Success is yours! (Now if only you knew
anything about fishing.)"



Test me with "test first / test second / test third".



Test first with "x captain / open case / i / ask captain for
the key / give beer to captain / open bench / x nets / get
nets / get coke / give coke to captain".



Test second with "s / open chest / drop nets / get glacier
ice / get nets / get glacier ice / g / put glacier ice in chest
/ get coke / put coke in chest".
```

```
Test third with "x coke / g / g / g / get coke / n / give
coke to captain / ask captain for key / open case / get
rods".
```

Example **Search and Seizure** *RB*

38

A smuggler who has items, some of which are hidden.

Suppose we want a character who carries hidden objects, but only while he is wearing his jacket. If we deprive him of this, his other possessions become known. Furthermore, if we ever search him, his possessions also become known, and can thereafter be mentioned by us.

```
"Search and Seizure"

Size is a kind of value. The sizes are small, medium,
and large. A thing has a size. A thing is usually small.

A thing can be licit or contraband. A thing is usually licit.

A thing can be discovered or secret. A thing is usually
secret.
```

Now, we want the character to be able to hide small things if he has some sort of concealing garment on. We also want to be able to see anything that the player has already found once, perhaps by using the >SEARCH PERSON command. So:

```
Rule for deciding the concealed possessions of
someone (called the suspect):
    if the particular possession is discovered, no;
    if the suspect wears something and the particular
```

```
possession is small, yes;
    otherwise no.
```

The following rule, borrowed from a later chapter, assures that any items that are ever mentioned to the player will be treated as known from then on:

```
Before printing the name of something (called
discovery): now the discovery is discovered.
```

```
The Customs House is a room. The smuggler is a man
in the Customs House. The smuggler wears a leather
jacket. He carries a bottle of Altairan rum, some
raspberries, and a laser pistol. The pistol is large. The
jacket is large. The rum and the raspberries are
contraband. The description of the smuggler is "He has
a bestubbled chin and a sinister eye."
```

```
The tourist is a woman in the customs house. The
description is "The type who walks off home with a
dozen contraband items in her pocket not because she
means to steal things but because she's too stupid to
understand that the law applies to herself." She wears a
tight-fitting dress. The dress is large. The tourist carries
a grapevine and an archaeological artifact. The
grapevine is large and contraband. The artifact is
contraband.
```

```
Report examining someone: say "[The noun] is [if the
noun is wearing something]wearing [a list of
unconcealed things worn by the noun] and [end
if]carrying [a list of unconcealed things carried by the
noun]."
```

```
Report examining someone who is concealing
something contraband:
    say "[The noun] looks nervous. You can just tell."
```

```
Report examining someone who is carrying an
unconcealed contraband thing:
    say "Your eye goes at once to [the list of unconcealed
contraband things carried by the noun] which [the noun]
is unable to hide";
    if an unconcealed licit thing is had by the noun, say ",
though [if the noun is female]s[end if]he also has [a list
of unconcealed licit things had by the noun]";
    say "." instead.
```

Notice that we can talk about what the smuggler wears, what he carries, and what he "has": things the smuggler has can be either worn or carried, so the phrase is useful if we don't care to make that distinction.

```
Instead of searching someone:
    say "[The noun] is revealed to be carrying [a list of
things carried by the noun]."
```

```
Instead of confiscating the dress:
    say "You are not allowed to perform strip-searches in
the public customs area."
```

```
Understand "confiscate [something]" as confiscating.
```

```
Confiscating is an action applying to one thing.
```

```
Check confiscating:
    unless the noun is had by someone who is not the
player:
        say "You can only confiscate other people's
possessions." instead.
```

```
Carry out confiscating:
    now the noun is carried by the player.
```

```
Report confiscating:
    say "Through the authority vested in you by the
power of the Sovereign of Centauri Proxima, you make
[the noun] your own."
```

```
Test me with "x smuggler / search smuggler / x
smuggler / confiscate jacket / x smuggler / confiscate
rum / confiscate pistol / x smuggler / confiscate
raspberries / x smuggler".
```

```
Test more with "x tourist / confiscate dress / confiscate
grapevine / x tourist / search tourist / confiscate artifact /
x tourist".
```

---

39 **ExampleVan Helsing** *RB*

A character who approaches the player, then follows him from room to room.

Suppose we want to write a character who tries to be in the same room as the player. We will do this by testing every turn whether the character's location and the player's location are the same; if the answer is no, the character will look for a path to the player's location, then try to move along that path. (We will learn more about finding paths and giving characters instructions later.)

The result will be that if the player ever moves to another location, the character will automatically pursue him.

```
"Van Helsing"
```

```
The Drawbridge is a room. North of the Drawbridge is
the Immensely Enormous Entry Hall. West of the Entry
Hall is the Vast Dining Area. North of the Vast Dining
Area is the Colossal Kitchen. The Spooky Guano-filled
```

```
Attic is above the Entry Hall.


Count Dracula is a man in the Attic.
```

In the following condition, we could also have written "if the location of Count Dracula is not the location", because "location" by itself is always understood to be the player's location. But it seemed better for clarity to write it this way:

```
Every turn:
    if the location of Count Dracula is not the location of
the player:
        let the way be the best route from the location of
Count Dracula to the location of the player, using doors;
        try Count Dracula going the way;
    otherwise:
        say "'Muhahaha,' says Count Dracula."


Test me with "z / z / n / w / e / u / z / d".
```

ExamplePrisoner's Dilemma *RB*

A button that causes a previously non-existent exit to come into being.

We can change the directions in the map in mid-game, though in practice this is rarely necessary. But suppose we do not want a door or any sign of a door to exist before the player takes some action, in this case pressing a button:

```
"Prisoner's Dilemma"


Challenger's Waiting Room is a room. "The challenge is
this: to wait as long as you can endure to do so in a
room with no features and no clock. If you wait longer
```

```
than all the other contestants, you win."


The button is fixed in place in the Challenger's Waiting
Room. "The only item in view is a black recessed
button."


Amid the Cheering Throng is a room.


Instead of pushing the button for the first time:
    change the east exit of the Challenger's Waiting
Room to Amid the Cheering Throng;
    change the west exit of the Cheering Throng to the
Challenger's Waiting Room;
    say "With a groan of gears, the east wall swings
open! If you've lost now, well, you've lost..."


Test me with "e / push button / e / w".
```

Our instructions about pushing the button will be further explained in the chapter on Actions, but the thing to note here is that we can "change (whatever) exit" in order to set or re-set map directions. Notice that we have to set both directions explicitly: changing the east exit of the Waiting Room does not automatically also change the west exit of Amid the Cheering Throng.

This allows greater flexibility in our games but does require an extra line or so of work.

Example The World of Charles S. Roberts *RB*

Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.

```
Wargaming is an ancient pursuit, but its modern form began as a
professional training exercise in 19th-century Prussian staff colleges;
since at least as early as H. G. Wells's "Little Wars" (1913) it has been a
hobby of "boys from twelve years of age to one hundred and fifty and for
```

that more intelligent sort of girl who likes boys' games and books." The free-form tabletop game used miniature figures and tape-measured movements, and remains the dominant form today. But in the mid-20th century, map grids on printed sheets gave the hobby a sudden new lease of life. They were easier to set up, more interesting to look at, cheaper to sell by mail-order. 1970s sales figures for "Strategy and Tactics", the leading US subscription-based wargame distributor, were very similar to those of Infocom's IF games in the 1980s. And like classical IF, the grid-based wargame parceled up a continuous world into locations.

Grids were initially square, as on a chessboard, but square cells have several disadvantages. Four directions of movement (N, E, S, W) is too few, yet allowing movement in the diagonal directions means allowing tanks to travel about 1.4 times faster northeast than they do north. Square grids also only conform cleanly to man-made landscape features such as buildings in one orientation, and they never fit hills well. (A compromise measure to fix this, cutting the squares into octagons to leave smaller diamond squares at corner intersections, has never caught on.)

But following Charles S. Roberts's American Civil War designs for Avalon Hill of 1958-61 (notably "Chancellorsville" and the second edition of "Gettysburg"), a hexagonal grid became the new standard. Each hexagon is the same distance from the centre of all six of its neighbours, which are at equal angular spacings; and clumps of hexagons fit the

shape of lakes, contoured hills, and so forth, much more naturally than clumps of squares do. Hexes also have a certain mystique - an air of "I don't belong in the children's department".

But hexes are tricky for IF, not least because English lacks words for "the direction 60 degrees around from front". Our cognitive view of the world tends to be square, perhaps because our two eyes both face front, in a direction at right angles to the plane of our arms, legs, pelvis and eyes. We reach out sideways at right angles to our walking. Even early hex-grid wargames called the cells "squares", though "hexes" eventually caught on. Still and all:

> "The World of Charles S. Roberts"

```
Forward is a direction. Forward has opposite backward.
Understand "f" as forward.
Backward is a direction. Backward has opposite
forward. Understand "b" and "back" as backward.
Forward left is a direction. Forward left has opposite
backward right. Understand "fl" as forward left.
Forward right is a direction. Forward right has opposite
backward left. Understand "fr" as forward right.
Backward left is a direction. Backward left has opposite
forward right. Understand "bl" as backward left.
Backward right is a direction. Backward right has
opposite forward left. Understand "br" as backward
right.
```

Now to forbid the use of the compass directions:

```
A direction can be hexagonal or squared-off. A direction
is usually squared-off. Forward, backward, forward left,
forward right, backward left and backward right are
hexagonal.
```

```
Before going a squared-off direction, say "In this
hexagonally-divided landscape, squared-off directions
are not allowed." instead.
```

A slight nuisance is that, with things as they are above, typing BACKWARD produces the response "Which do you mean, backward,

backward left or backward right?" To avoid that silly question, we write:

```
Does the player mean going backward: it is very likely.
Does the player mean going forward: it is very likely.
```

And now a clump of 37 hexes, in six columns of six or seven rooms each. There are many ingenious ways we could put this map together automatically, but instead we will take a deep breath and write:

```
E1 is forward of E2. "Open farmland." E2 is forward of
E3. "The edge of woods." E3 is forward of E4. "Deep
woodland." E4 is forward of E5. "Deep woodland." E5 is
forward of E6. "The rear edge of woods." E6 is forward
of E7. "The start of a road leading forward right." E7 is a
room. "Grassland."
```

```
F1 is forward of F2. "The edge of farmland." F2 is
forward of F3. "The edge of woods." F3 is forward of F4.
"Clearing in woods." F4 is forward of F5. "Deep
woodland." F5 is forward of F6. "A road runs backward
left to forward right." F6 is a room. "The edge of
grassland."
```

```
G1 is forward of G2. "Grassland." G2 is forward of G3.
"The edge of farmland." G3 is forward of G4. "A copse
of trees." G4 is forward of G5. "The backward edge of
woodland." G5 is forward of G6. "A bend in the road,
from backward left to backward right." G6 is forward of
G7. "Open farmland." G7 is a room. "Open farmland."
```

```
H1 is forward of H2. "Grassland, bordered by a hedge to
the right." H2 is forward of H3. "The edge of farmland,
with a hedge to forward right." H3 is forward of H4. "A
copse of trees." H4 is forward of H5. "Open farmland."
H5 is forward of H6. "A passing place on the road,
which bends forward left to forward right." H6 is a room.
"Open farmland."
```

I1 is forward of I2. "The end of a forward road, blocked by hedges on all sides except backward." I2 is forward of I3. "A straight road runs forward to backward, with long hedges to left and right." I3 is forward of I4. "A straight road runs forward to backward, alongside a long hedge to right." I4 is forward of I5. "A straight road runs forward to backward, alongside a long hedge to right." I5 is forward of I6. "Where three roads, forward, backward left and backward right, meet. Forward right is a thick hedge." I6 is forward of I7. "Open farmland." I7 is a room. "Open farmland."

J1 is forward of J2. "Dense woodland, with a hedge to left." J2 is forward of J3. "Grassland, with a hedge to left." J3 is forward of J4. "The edge of farmland, with a hedge to left." J4 is a room. "Open farmland, with a long hedge blocking movement forward left, backward left or backward." J5 is forward of J6. "A road running forward left to backward right, alongside a hedge." J6 is a room. "Open farmland."

F1 is forward right of E2 and backward right of E1. F2 is forward right of E3 and backward right of E2. F3 is forward right of E4 and backward right of E3. F4 is forward right of E5 and backward right of E4. F5 is forward right of E6 and backward right of E5. F6 is forward right of E7 and backward right of E6.

G1 is forward right of F1. G2 is forward right of F2 and backward right of F1. G3 is forward right of F3 and backward right of F2. G4 is forward right of F4 and backward right of F3. G5 is forward right of F5 and backward right of F4. G6 is forward right of F6 and backward right of F5.

H1 is forward right of G2 and backward right of G1. H2 is forward right of G3 and backward right of G2. H3 is forward right of G4 and backward right of G3. H4 is forward right of G5 and backward right of G4. H5 is forward right of G6 and backward right of G5. H6 is forward right of G7 and backward right of G6.

```
I3 is forward right of H3 and backward right of H2. I4 is
forward right of H4 and backward right of H3. I5 is
forward right of H5 and backward right of H4. I6 is
forward right of H6 and backward right of H5.
```

```
J5 is forward right of I6 and backward right of I5. J6 is
forward right of I7 and backward right of I6.
```

And now we have a hexagonally-gridded world. Route-finding will work; prepositional forms like "to be mapped backward left of" exist, just as they should; and in general these directions are just as good as the square ones. (The only thing which doesn't look good is the Index map, where Inform is just unable to draw a picture because it assumes a square grid. But that has no effect on play.)

The landscape is much easier to navigate with a little diagram:

```
To say legend (D - direction):
    let destination hex be the room D from the location;
    if the destination hex is nothing, say " ";
    otherwise say the destination hex.
```

```
Carry out looking:
    say "[fixed letter spacing] \ [legend forward] /[line
break][legend forward left] ---- [legend forward right][line
break] / \[line break]--< [location] >--[line break] \ /[line
break][legend backward left] ---- [legend backward right]
[line break] / [legend backward] \[variable letter spacing]
[line break]".
```

And finally:

```
The player is in I5.
```

```
Test me with "f / forward / backward left / bl / br / br / f".
```

ExampleFore *RB*

Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

Suppose we want to understand shipboard directions, but only when the player is aboard a vessel.

"Fore"

Section 1 - Procedure

The starboard is a direction. The starboard has opposite port. Understand "s" as starboard when the location is nautical.

The port is a direction. The port has opposite starboard. Understand "p" as port when the location is nautical.

The fore is a direction. The fore has opposite aft. Understand "f" as fore when the location is nautical.

The aft is a direction. The aft has opposite fore. Understand "a" as aft when the location is nautical.

Does the player mean going a nautical direction when the location is nautical: it is very likely.

Index map with fore mapped as north. Index map with aft mapped as south. Index map with port mapped as

```
west. Index map with starboard mapped as east.
```

And we can even add new ways to talk about the ways things are mapped, borrowing from the Relations chapter. The following will allow us to use "is abaft of" as well as "is aft of":

```
[The verb to be abaft of means the mapping aft relation.]
```

Now, to prevent the player from using NORTH onboard ship, or AFT on land:

```
A room can be nautical or earthbound. A room is usually
not nautical. A direction can be nautical or earthbound.
A direction is usually not nautical. Starboard, port, fore,
aft, up, down, the inside and the outside are nautical.

Before going a nautical direction when the location is
not nautical, say "Nautical directions can only be used
on board ship."

Before going an earthbound direction when the location
is nautical, say "Compass directions make no sense on
board ship, but you can use [list of nautical directions]
instead." instead.

Section 2 - Scenario

The Fish Room is aft of the Spirit Room. Starboard of
the Fish Room is the After Powder Magazine. The
Bread Room is aft of the After Powder Magazine.

The Fish Room, the Spirit Room, the Bread Room, and
the After Powder Magazine are nautical.
```

The description of the Fish Room is "Absurd quantities of salt fish are kept here, and periodically visited by the cook or someone serving him. It is otherwise an unexceptional little chamber, so far below the waterline that there are no portholes and no external light of any kind. [paragraph break]A narrow doorway leads forward into the Spirit Room, and the After Powder Magazine is starboard."

The description of the Spirit Room is "Despite its ghostly name, this is little more than a closet down at the very navel of the ship, in which alcohol is kept: both for purifying wounds and for drinking. Under normal circumstances there is a guard posted here at every hour, lest anyone take to raiding the larder. The current absence of the guard marine strikes you as a very bad sign indeed. [paragraph break]The only way out is aft."

The description of the Bread Room is "The Bread Room is not only tiny from side to side and front to back: it is also about half the height of a proper room, and the floor slopes up very steeply with the curve of the hull. [paragraph break]What is kept here would not, on land, be dignified by the name of bread: it's hard tack, punishing to the teeth, dry on the tongue, and usually a home to weevils before half the journey is done. [paragraph break]More headroom, and access to the rest of the ship, lies fore through the After Powder Magazine."

The description of the After Powder Magazine is "Kept in near darkness because no one with any sense would bring a naked flame down here: when necessary, it can be lit with a single small lantern made of very thick glass and sealed to keep the sparks within. Sacks of powder are passed up into the higher levels of the ship by the scrubby little boys called 'powder monkeys' -- but none such are here now."

Test me with "north / aft / fore".

Chapter 4: Kinds

roperties; *§4.9. Using new kinds of value in properties*; *§4.10. Conditions of things*; *§4.11. Default value*

Contents of *Writing with Inform*
Chapter 3: Things
Chapter 5: Text
Indexes of the examples

§4.1. New kinds

Values are to Inform what nouns are to English sentences. They represent numbers, times of day, pieces of text, places, people, doors, and so on. Because they have such an enormous variety, and because we often want to talk about what some of them have in common, we need a way to sort all of these different ideas out. That's the main aim of Inform's concept of "kind".

Every value has a kind. The kind of 10 is "number"; the kind of 11:30 PM is "time"; the kind of "jinxed wizards pluck ivy from my quilt" is "text"; and so on. The Kinds index panel shows the kinds present in the current Inform project, which will always include a wide range of built-in kinds, and usually also some new ones created in that project.

Some kinds are more general than others. For example, if we write:

```
Growler is an animal in the Savannah.
```

then Growler is an "animal", which is a kind of "thing", which is a kind of "object". When we talk about "the" kind of Growler, we mean "animal", the most specific one, but actually he belongs to all of those kinds.

As we see from this example, kinds have a whole hierarchy. Some are specialised versions of other kinds; some are not. Browsing the Kinds index shows that Inform builds its model world out of "objects". (That's really what objects are: "object" is a kind of value used to make the ingredients of the model world.) The objects fall into four fundamental kinds, "room", "thing", "direction" and "region", and "thing" is more subdivided still.

All the same, Inform starts out with a fairly simple arrangement. Because taxonomy - setting up kinds for everything - is so difficult, and depends so much on what you want it for, Inform creates relatively few kinds in advance: it has "animal" built in, but

not "mammal" or "Bengal tiger". When we need more, we must make them ourselves. Let's see how we might invent these new kinds. The easy one is:

```
A Bengal tiger is a kind of animal.
```

Given that, we can then write:

```
Growler is a Bengal tiger in the Savannah.
```

That's easy enough. Adding "mammal" now looks awkward, though, because it seems to belong in between the two. All Bengal tigers are mammals, but not all animals are. But Inform can sort this out:

```
A mammal is a kind of animal. A Bengal tiger is a kind of mammal.
```

If we look at the Kinds index, we should indeed see a hierarchy:

```
object > person > animal > mammal > Bengal tiger
```

though the diagram is laid out as a sort of tree, just as zoologists do.

As another example, it may seem curious that Inform doesn't usually think "man" is a kind of "animal". This is really because, in English, people don't usually expect something like "if an animal is in the garden" to be true when it's only the Revd Mr Beebe going for a walk. People expect the word "animal" not to mean a human being, despite our common genetic ancestry. But if we want to rearrange Inform's default ideas, we can do that quite easily:

```
A man is a kind of animal. A woman is a kind of animal.
```

Or indeed we could say:

```
A human being is a kind of mammal. A man and a woman are kinds
of human being.
```

While this is an ideal way to make new kinds for the model world, we are more

restricted in what we can do outside of objects. For instance,

```
A glob is a kind of number.
```

isn't allowed. The numbers are fixed and all exist already; they aren't like Bengal tigers which we can simply imagine, and fill the world with. "Number" is not a concept we can specialise any further. But what we can do is to invent entirely new concepts, like so:

```
A distance is a kind of value.
```

We will see more of these later. (This isn't specialising anything - "value" is a sort of everything-category, and is too big and vague to be a kind.)

Start of Chapter 4: Kinds

Back to Chapter 3: Things: §3.26. Directions

Onward to §4.2. Using new kinds

Example 43: ★★ **Vouvray** Adding synonyms to an entire kind of thing.

---

§4.2. Using new kinds

This seems a good point to see what we can do with new kinds. Here we invent a new kind to provide a new sort of room:

```
A dead end is a kind of room.
```

Any dead end that we make is also a room, so it has all of the properties and behaviour of a room. For instance, every room is either "lighted" or "dark", and the default is to be lighted. But we can reverse that convention for dead ends, and we can also fill in some other properties:

```
A dead end is a kind of room with printed name "Dead End" and
```

```
description "This is a dead end. You'll have to go back the way you
came." A dead end is usually dark.
```

```
The Undertomb is a dark room. East is a dead end. South is a dead
end with printed name "Collapsed Dead End". Northwest is a dead
end called the Tortuous Alcove.
```

```
In the Undertomb is the candle lantern. It is lit.
```

As a result of this, three different rooms adjoin the Undertomb, all dead ends. This is much more concise than spelling them out one at a time would be.

Inform often doesn't mind in what order it is told about the world, but it may need to know the name of a kind before that kind can be used. For example,

```
A coffer is a kind of container. In the Crypt is an open coffer.
```

makes sense to Inform and results in the creation of a new thing, just called "coffer" in the absence of any other name to give it, whose kind is "coffer" and which is initially open. Whereas if Inform reads:

```
In the Crypt is an open coffer.
```

without knowing that "coffer" is a kind, it simply makes a thing called "open coffer" (and which is not a container). Inform has to be careful like this: English is simply too overflowing with multiple meanings. An "open railway ticket", for instance, is not a "railway ticket" that one can put objects into.

Start of Chapter 4: Kinds

Back to §4.1. New kinds

Onward to §4.3. Degrees of certainty

§4.3. Degrees of certainty

When we write:

```
A dead end is usually dark.
```

we are saying that it will be dark rather than lighted unless we should specify otherwise. So it would be fine to add:

```
The Tortuous Alcove is lighted.
```

because although dead ends are usually dark, this one is evidently an exception. On the other hand, if we had originally written

```
A dead end is always dark.
```

then Inform would not have permitted any exception to be made, and would have reported a problem if we had tried to make the Tortuous Alcove lighted. Besides "usually" and "always", we can also employ "seldom" and "never", which are their negatives. Thus, "never lighted" means the same as "always dark".

Start of Chapter 4: Kinds

Back to §4.2. Using new kinds

Onward to §4.4. Plural assertions

Example 44: **Odin** Replacing "You see nothing special..." with a different default message for looking at something nondescript.

Example 45: **Something Narsty** A staircase always open and never openable.

---

§4.4. Plural assertions

As the following examples show, sentences can make several assertions at once by using the plural. Suppose we have defined a kind called "high-up fixture", for instance like so:

```
A high-up fixture is a kind of thing. A high-up fixture is usually fixed in
place.
```

Then the following sentence creates two such objects and puts them in their place:

```
The high shelf and the skylight window are high-up fixtures in the
Lumber Room.
```

since it is equivalent to saying:

```
The high shelf is a high-up fixture. The skylight window is a high-up
fixture. The high shelf is in the Lumber Room. The skylight window is
in the Lumber Room.
```

Such plurals are allowed in almost any context, and we could even define two kinds at once:

```
Bucket and basket are kinds of container.
```

Inform constructs plurals by a form of Conway's pluralisation algorithm, which is quite good - for example, it gets oxen, geese (but mongooses), sheep, wildebeest, bream, vertebrae, quartos, wharves, phenomena, jackanapes and smallpox correct. But English is a very irregular language, and multiple-word nouns sometimes pluralise in unexpected ways. So we sometimes need to intervene:

```
A brother in law is a kind of man. The plural of brother in law is
brothers in law.
```

We are allowed to define more than one plural for the same singular text, and for the names of things, rooms or kinds, all alternative versions will be used interchangeably. (For instance, Inform defines both "people" and "persons" as plurals of "person".)

Start of Chapter 4: Kinds

Back to §4.3. Degrees of certainty

Onward to §4.5. Kinds of value

Example 46: ★★★ **Get Me to the Church on Time** Using

kinds of clothing to prevent the player from wearing several pairs of trousers at the same time.

§4.5. Kinds of value

So much for making new and more specialised kinds of object - for example, new kinds of room, or new kinds of animal. This allows us to describe the physical world in elegant ways, but what about concepts which aren't so physical?

Without getting into philosophy, we can probably agree that numbers like 1, 2, 3, ..., and texts like "Jackdaws love my big sphinx of quartz", are not physical. Inside Inform, those are values, but not objects. Inform already has a good stock of this sort of concept built in, so it may not immediately seem clear why we need to create new ones. But in fact this is very useful. To describe the physical world, we need concepts like (for example) distance and brightness. We want to say that two armchairs are 12 feet apart, or that a given light-bulb is very dim. Here, "twelve feet" and "very dim" are clearly not physical objects; they need to be values, but not objects.

As these two examples suggest, sometimes we want a quantitative way to measure things, sometimes not. Thomas Hardy, in his novel "The Return of the Native", writes:

> When he drew nearer he perceived it to be a spring van, ordinary in shape, but singular in colour, this being a lurid red.

Hardy doesn't tell us that the wavelength of the light is 700nm, he tells us that the colour is "lurid red", and we understand. Later in the same chapter, Hardy writes:

> The loads were all laid together, and a pyramid of furze thirty feet in circumference now occupied the crown of the tumulus.

and now we do have a quantitative measurement: thirty feet. This is how people write about the world, and how they read about it. So Inform needs to provide both sorts of measurement.

(a) Here is a qualitative example. Suppose we would like a candle lantern to burn down, gradually diminishing in brightness. Then we'll need a way to talk about the current strength of the flame, but only in vague terms. Here goes:

```
Brightness is a kind of value. The brightnesses are guttering, weak,
```

```
radiant and blazing.
```

"Brightness" is now a kind of value on a par with (for instance) "number" or "text". There are only four possible values, named as above. Kinds of value like this, where there are just a few named possibilities, are extremely useful, as we'll see.

(b) Now a quantitative example:

```
Weight is a kind of value. 1kg specifies a weight.
```

The difference here is not the way we create the kind, but the way we tell Inform what the possible values are. Instead of a list, we teach Inform some notation. As a result, "26kg" is now a value, for instance. Quantitative kinds like this are sometimes called "units", because - as in this example - they're often units in the sense of measuring things. Many Inform projects never need units, but they can still be very useful, and they're described in detail in the chapter on "Numbers and Equations".

Start of Chapter 4: Kinds
Back to §4.4. Plural assertions
Onward to §4.6. Properties again

§4.6. Properties again

So now we have seen two fundamental ideas: "value" and "kind". We have seen how to make a source text which refers to many values - for example, Growler the Bengal tiger, 23kg, "Collapsed Dead End", blazing, 7, all values of different kinds.

But we don't just want a way to refer to values, we want to lay out facts about them. Inform understands two sorts of fact, which it calls properties and relations. Properties are about single values in isolation: Growler is male. Relations are about how values interact with each other: Growler likes Bambi. (Or would like to eat Bambi, anyway.) Relations are really the central organising idea of Inform, and we've seen them many times already:

```
Growler is in the Savannah.
```

expresses a relation called "containment" between Growler and the Savannah. Much more about this in the chapter on Relations: for now, let's go back to the simpler idea of properties.

In Inform terms, a "property" is any fact about a value (other than its kind) which the

author is allowed to choose. For example,

```
Growler is an animal. Growler is male. The description of Growler is
"What immortal hand or eye could frame thy fearful symmetry?".
```

The first of these sentences talks about Growler's kind, but the other two sentences tell Inform about his properties. Properties are divided into either/or properties - "male" versus "female" - and value properties - such as the description of something, which can be any text.

The Kinds index shows which kinds of object are allowed to have properties. Every object is, so there's no problem with Growler. In general, if Inform can find a sensible way to store properties, then it will allow them. But it won't allow (for example) properties of numbers. There are only a finite number of Bengal tigers in the world (fewer than three thousand, alas), so Inform can easily store individual description texts for each one of them. But there are an infinite range of numbers. (Inform does allow adjectives like "even" or "odd" to be used about number - saying "if 7 is odd" is fine, for example - but they are not properties in the Inform sense, because the author doesn't get to choose. The author can choose whether Growler is a male or female tiger, but not whether 7 is an even or an odd number.)

Start of Chapter 4: Kinds
Back to §4.5. Kinds of value
Onward to §4.7. New either/or properties

§4.7. New either/or properties

Properties can't be handed out completely freely. In the previous chapter, we saw that we were allowed to make a chair "portable" and to make a room "dark". But if we try this the other way round, Inform produces a Problem message. This is because every property must be created in a way which lays out what values are allowed to have it. The Standard Rules built into Inform say that

```
A thing can be fixed in place or portable.
```

and as a result it won't allow "The Savannah is portable" because the Savannah is a room, not a thing.

We must do the same. To go back to our example "dead end" kind:

```
A dead end is either secret or ordinary.
```

This creates just one new property, not two. The names are taken as the two states of a single either/or property: secret means not ordinary, ordinary means not secret. Alternatively, we could just say:

```
A dead end can be secret.
```

in which case the opposite of "secret" would be "not secret".

Now we have a property which can be given to any value of the kind "dead end". We're also free to add to the definitions of kinds which already exist, including those built into Inform: for instance,

```
A room is either indoors or outdoors.
```

If we make the above definitions then all dead ends will be "ordinary" and all rooms "outdoors" unless the source text says otherwise. That is, in the absence of other information it's assumed that an either/or property is not true. We could reverse by writing:

```
A dead end is usually secret. A room is usually indoors.
```

A property can be used by several kinds at once. For example, the built-in either/or property "open" is used by both doors and containers, even though door isn't a kind of container and container isn't a kind of door. In fact, although it's more usual to declare properties for whole kinds, they can actually be given to single values one at a time, if we like:

```
The umbrella is carried by the player. The umbrella can be open.
```

And now the umbrella, which is a thing and not a door or container, can also have the property.

Start of Chapter 4: Kinds
Back to §4.6. Properties again

Onward to §4.8. New value properties

Example 47: ★★ **Change of Basis** Implementing sleeping and wakeful states.

§4.8. New value properties

So much for either/or properties. Now we move on to properties which have values attached. The same principles apply, but the wording is different. For example,

```
A dead end has some text called the river sound. The river sound of
a dead end is usually "a faint whispering of running water". The
Tortuous Alcove has river sound "a gurgle of running water".
```

The property "river sound" is now applicable only to dead ends, so we would not be allowed to talk about "the river sound of the Savannah", say. Moreover, it's required to hold a piece of text. If we tried the following:

```
The river sound of the Tortuous Alcove is 7.
```

...then Inform would object, because the number 7 is the wrong kind of value to go into the "river sound" property. If we need a numerical property, we can try this instead:

```
A dead end has a number called the difficulty rating. The Tortuous
Alcove has difficulty rating 7.
```

Suppose that we were to add:

```
The Exquisitely Narrow Defile is a dead end.
```

The Defile must have a river sound, of course, because we said that every dead end would have one. We haven't said what that river sound will be, but Inform can work it out, because we did say this:

```
The river sound of a dead end is usually "a faint whispering of
running water".
```

If there are no instructions at all about the value of a property, Inform fills in the default value of the appropriate kind - in this case, it would be a blank text. (A table of the kinds which can be used for properties, and their default values, can be found in the Kinds index.)

Start of Chapter 4: Kinds

Back to §4.7. New either/or properties

Onward to §4.9. Using new kinds of value in properties

Example 48: ★ **Would you...?** Adding new properties to objects, and checking for their presence.

Example 49: ★★ **Straw Boater** Using text properties that apply only to some things and are not defined for others.

---

§4.9. Using new kinds of value in properties

It turns out to be very useful to create a new kind of value, and then create a property to hold it. So useful, in fact, that Inform provides two shortened forms for doing so. Here is the first, and the second (making "conditions") is in the section following.

Suppose we go back to our example of the candle lantern whose brightness we have to measure. It's clear that what we want to do is to define:

```
Brightness is a kind of value. The brightnesses are guttering, weak,
radiant and blazing.
```

And now we can use the technique of the previous section:

```
The lantern has a brightness called the flame strength. The flame
strength of the lantern is blazing.
```

This works very nicely. The "flame strength" property is now only allowed to have one

of four values we allowed: guttering light, weak light, radiant light or blazing light. So we have succeeded in recording our measurement.

But it seems artificial to call the brightness of the lantern "flame strength", when we could instead simply call it "brightness". Much simpler to write:

```
The lantern has a brightness. The lantern is blazing.
```

Now "brightness" is the name of both the property and the kind of value. What's particularly nice is that we can now use the names of the possible brightnesses - "weak", "blazing" and so on - as adjectives. Inform knows that "The lantern is blazing" must be talking about the brightness property, because "blazing" is a brightness.

Now we can improve our dead ends:

```
A dead end is a kind of room with printed name "Dead End" and
description "This is a dead end, where crags in the uneven rock are
caught by the [brightness of the lantern] flame you hold aloft. Despite
[river sound] there is no sign of the stream." A dead end is usually
dark.
```

The "[brightness of the lantern]" is printed not as literal text, but as whatever the brightness currently is. (The square brackets mark it as what is called a text substitution, which will be the subject of the next chapter.) So we get something like this:

```
This is a dead end, where crags in the uneven rock are caught by the
blazing flame you hold aloft. Despite a faint whispering of running
water there is no sign of the stream.
```

So now we have a lantern, which has a brightness as a property. But we can build on this further if we like. A brightness such as "guttering" is a value, so it can have properties in its own right. That can be quite useful, in fact:

```
A brightness can be adequate or inadequate. A brightness is usually
adequate. Guttering is inadequate.
```

This is convenient because it divides up the brightnesses:

```
The player carries a book. The description of the book is "[if the
brightness of the lantern is adequate]Many secrets are now yours.
[otherwise]No, the print's too tiny by this awful light."
```

And while we're at it, let's give each brightness its own corresponding temperature:

```
Temperature is a kind of value. 100C specifies a temperature.
```

```
A brightness has a temperature. The temperature of a brightness is
usually 700C. The temperature of blazing is 1400C. The temperature
of radiant is 1100C.
```

```
The description of the lantern is "The lantern shines with a flame at
[temperature of the brightness of the lantern]."
```

(Candle flames are hotter than most people think.)

See **Text with substitutions** *for more on varying what is printed*

Start of Chapter 4: Kinds

Back to §4.8. New value properties

Onward to §4.10. Conditions of things

Example 50: **The Undertomb 1** A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

Example 51: **The Undertomb 2** Flickering lantern-light effects added to the Undertomb.

Example 52: **The Crane's Leg 1** A description text that automatically highlights

the ways in which the object differs from a standard member of its kind.

Example 53: ★★★ **Signs and Portents** Signpost that points to various destinations, depending on how the player has turned it.

---

§4.10. Conditions of things

Now for an even more abbreviated way to create a new kind of value, and at the same time create a property to hold it. Suppose we have something, say a wine cask, which we know is always in one of three different states. We can write:

```
The cask is either customs sealed, liable to tax or stolen goods.
```

This is just like our example of the lantern having possible brightnesses, but it's quicker to do, because we don't need to create or name the kind of value. (The trade-off is that we can't use it for anything else as well.)

Initially the cask will be "customs sealed", the first value we gave. We could now write, for instance,

```
The description of the cask is "A well-caulked Spanish wine cask.
[if liable to tax] It really is a shame to have to pay duty on it!"
```

Or, as a second example, here we're going to allow a whole kind to have the property, not just a single object:

```
Colour is a kind of value. The colours are red, green and white.
A colour can be bright, neutral or flat. Green is neutral.
```

Now in fact these properties are not anonymous: Inform has worked out names for them, even though we didn't give any. The usual arrangement is that the name is the name of the object with the word "condition" tacked on: for instance, "cask condition". So we could write:

```
The printed name of the cask is "wine cask ([cask condition])".
```

so that sometimes this would be "wine cask (liable to tax)", sometimes "wine cask (stolen goods)" and so on.

But only usually, because we might need to define several different conditions of the same thing, and then the names would collide. For instance, suppose we write:

```
A fruit is a kind of thing. A fruit can be citrus, berry, melon, or pome.
```

This makes a property and a kind of value each called "fruit condition". But now suppose we add that:

```
A fruit can be unripened, ripe, overripe, or mushy.
```

This is a quite unrelated property - a fruit could have any combination of these two properties, in fact. Left to itself, Inform will call the second one "fruit condition 2", which isn't really ideal if we ever do need to refer to it in other source text. So we are also allowed to give these conditions names of our own choosing:

```
A fruit can be unripened, ripe, overripe, or mushy (this is its
squishiness property).
```

And now the resulting property and kind of value would be called "squishiness".

Start of Chapter 4: Kinds

Back to §4.9. Using new kinds of value in properties

Onward to §4.11. Default values of kinds

---

§4.11. Default values of kinds

Just about every kind has a "default value". Inform needs this when it knows that something has to be a value of a given kind, but it hasn't been told what the value is. For example, in the previous chapter we saw that every thing has a "description" text, but we also created plenty of things without describing them. So if Inform reads

```
The conference pear is in the bowl.
```

and it isn't told anything else about the pear, what should it set the description of the pear to?

The answer is that Inform knows the description has to be a value of the kind "text", so it uses the default value of "text". Not very interestingly, this is just the blank text "".

Being uninteresting is exactly the idea, of course. The default number is 0, for instance. (Default values are tabulated in the Kinds index.)

It's sometimes useful to be able to refer to the default value of a kind without having to spell out what this is (especially if the kind is something obscure, or we're trying to write a rule for an extension which has to work in situations we don't fully know about).

default value of (name of kind) **... value**

Produces the default value of the kind named. Examples:

```
The silver repeater is here. "You catch sight of a silver
repeater watch, hands immobile at [default value of time]."
```

produces the output:

```
You catch sight of a silver repeater watch, hands immobile
at 9:00 am.
```

because nine in the morning is the default time in Inform. If we have:

```
Brightness is a kind of value. The brightnesses are
guttering, weak, radiant and blazing.
```

then "default value of brightness" is guttering, the first brightness created. When it comes to kinds of object, we sometimes have to be a little careful. For example,

```
default value of room
```

is always going to be fine (it's always the first room created in the source text). But

```
default value of vehicle
```

would produce a Problem message if there were no vehicles in the world.

Start of Chapter 4: Kinds
Back to §4.10. Conditions of things
Onward to §4.12. Values that vary

§4.12. Values that vary

Sometimes a value important to the simulated world will not naturally belong to any thing or room, and should not be kept in a property. In fact, we have seen a value that varies already: "location", which holds the room in which the story is presently taking place. Here's how we might make a new one:

```
The prevailing wind is a direction that varies. The prevailing wind is southwest.
```

Or "which varies" would also be allowed, as would the more traditional computing term "variable":

```
The prevailing wind is a direction variable. The prevailing wind is southwest.
```

A briefer way to do this is to use the word "initially", which alerts Inform to the possibility that the value will change in future:

```
The prevailing wind is initially southwest.
```

This creates the variable and gives it an initial value all in one sentence.

It's not compulsory to give an initial value. If we do not, Inform will use the default value for its kind. (See the table in the Kinds index.) For example, writing just

```
The grand tally is a number that varies.
```

will start it at the value 0, because that's the default value for numbers.

We can have variables of any of the kinds of value, including new ones, but should watch out for a potential error. If we write:

```
The receptacle is a container that varies.
```

in a world which has no containers at all, Inform will object, because it will be unable to put any initial value into the receptacle variable. A similar complaint will be made if we write:

```
Colour is a kind of value. The fashionable shade is a colour that
varies.
```

without ever having defined any colours. Something else we are not permitted is:

```
The receptacle is an open container that varies.
```

because the openness of a given container may change during play, so that the value in the variable might suddenly become invalid even though the variable itself had not changed.

As a final note on kinds, when Inform reads something like this:

```
Peter is a man. The accursed one is initially Peter.
```

it has to make a decision about the kind of "accursed one". Peter is a "man", so that seems like the right answer, but Inform wants to play safe in case the variable later needs to change to a woman called Jane, say, or even a black hat. So Inform in fact creates "accursed one" as an object that varies, not a man that varies, to give us the maximum freedom to use it. If we don't want that then we can override it:

```
Peter is a man. The accursed one is initially Peter.
The accursed one is a man that varies.
```

thus telling Inform exactly what is intended.

Start of Chapter 4: Kinds

Back to §4.11. Default values of kinds

Onward to §4.13. Values that never vary

Example 54: ★★★ **Real Adventurers Need No Help** Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

§4.13. Values that never vary

It's sometimes useful to name even values which don't change. For example, suppose the story involves driving, and the same speed limit value comes up in many places. Rather than typing "55" (say) every time it comes up, we might prefer to write:

```
The speed limit is always 55.
```

at the start of the source text, and then talk about "the speed limit" every time we would otherwise have typed "55". Just as the word "initially" alerts Inform that we want the named value to change during play, the word "always" tells it that we don't.

This might seem pointless, because "speed limit" only means the same thing as "55" and takes more typing. But there are two reasons why authors might want to use this feature anyway. One is that it's easier for a human reader to understand the significance of a line like:

```
if the speed is greater than the speed limit, ...
```

Another is that it makes it easier to change our minds about the value, because if we decide we want 70 as the limit and not 55, we only need to make one change at the

start of the source text:

```
The speed limit is always 70.
```

which is much easier than combing through a long source text trying to find many individual things which need changing.

"Speed limit" is then a number constant. Any attempt to set this elsewhere, or change its value, will result in a Problem message, and moreover it can be used in contexts where only constant values are allowed. For example,

```
The generic male appearance is always "He is a dude."

Trevor is a man. The description of Trevor is the generic male
appearance.
```

means that the SHOWME TREVOR testing command produces, among other data:

```
description: "He is a dude."
```

Start of Chapter 4: Kinds
Back to §4.12. Values that vary
Onward to §4.14. Duplicates

§4.14. Duplicates

Although it is only useful to a limited extent, we can make any number of copies of something:

```
"Polygons"

A shape is a kind of thing. A square is a kind of shape. A triangle is a
```

```
kind of shape.
```

```
The Geometry Lab is a room. In the Geometry Lab are three triangles
and two squares.
```

The description "three triangles" makes three identical things, each of the kind "triangle", and similarly for the squares. When the above is compiled, the player can type TAKE TWO TRIANGLES or TAKE ALL THE TRIANGLES and so forth.

Four caveats. Firstly, a counted-out description like "two squares" is only allowed if it combines a number with the name of a kind which is already known (perhaps modified with adjectives, so "two open doors" is fine). If we say:

```
Two circles are in the Lab.
```

without having defined "circle" as a kind in advance, then only a single object will be created - whose name is "two circles". (This is because many natural names start with numbers: "six of clubs", for instance, referring to a single playing card, or "12 Hollywood Close" meaning a single house. We wouldn't want such names to be misinterpreted.)

The second caveat is that excessive duplication is expensive in memory and running time. It is perfectly legal to say

```
In the Lab are 75 triangles.
```

but the resulting story may be a little sluggish: and Inform draws the line at 100, refusing to create more duplicates than that in any single place. If we really need more than about fifty duplicated objects - say, a tombola containing raffle tickets numbered 1 to 1000 - it is usually better to find some less literal way to simulate this: for instance, only having a single raffle ticket, but with a randomly chosen number on it.

If there are very many items in the same place, commands like TAKE ALL and DROP ALL may mysteriously not quite deal with all of them - this is because the parser, the run-time program which deciphers typed commands, has only limited memory to hold the possibilities. It can be raised with a use option like so:

```
Use maximum things understood at once of at least 200.
```

(The default is, as above, 100. Note the "at least".)

Thirdly, note that Inform's idea of "identical" is based on what the player could type in a command to distinguish things. In a few cases this can make items unexpectedly identical. For example:

```
The Lab is a room. A chemical is a kind of thing. Some polyethylene
and polyethylene-terephthalate are chemicals in the Lab.
```

results surprisingly in "You can see two chemicals here", because the run-time system truncates the words that are typed - POLYETHYLENE and POLYETHYLENE-TEREPHTHALATE look like the same word in a typed command. So Inform decides that these are indistinguishable chemicals. Typically words are truncated after 9 letters, though (unless the Glulx setting is used) punctuation inside a word, such as an apostrophe, can make this happen earlier. The best way to avoid trouble is simply to use more easily distinguishable names. For example:

```
Some polyethylene and polyethylene terephthalate are chemicals in
the Lab.
```

works fine, because now only one chemical can be called TEREPHTHALATE, and that means they can be distinguished.

Finally: numbers up to twelve may be written out in words in the source text, but larger ones must be written as numerals. So "twelve" or "12", but "13" only.

Start of Chapter 4: Kinds

Back to §4.13. Values that never vary

Onward to §4.15. Assemblies and body parts

Example 55: ★★★ **Early Childhood** A child's set of building blocks, which come in three different colours - red, green and blue - but which can be repainted during play.

§4.15. Assemblies and body parts

In the previous chapter, we saw that it was possible to make sub-parts of things. For

instance,

```
The white door is in the Drawing Room. The handle is part of the
white door.
```

creates a door with an attached handle. But what if we want to say that not just this door, but every door, should have a handle? To do this we first need to create a kind called "handle", since there will clearly need to be many handles. The solution is:

```
A handle is a kind of thing. A handle is part of every door.
```

"Every" is a loaded word and best used sparingly. A sentence like "A handle is part of every handle" would, if taken literally, mean that a handle takes forever to make and is never finished. Inform will reject this, but the moral is clear: we should think about what we are doing with "every".

We will usually want to work with smaller collections - not literally every room, but with a whole set of them all the same. We can do that like so:

```
A silver coin is a kind of thing. A banking room is a kind of room. Five
silver coins are in every banking room.
```

The effect of sentences like these is to make what we might call "assemblies" instead of single things. When a banking room is created, so are five more silver coins; when a door is created, so is another handle. Such sentences act not only on items created later on in the source text, but also on all those created so far.

This is especially useful for body parts. If we would like to explore Voltaire's suggestion that history would have been very different if only Cleopatra's nose had been shorter, we will need noses:

```
A nose is a kind of thing. A nose is part of every person.
```

Of course, if we make an assembly like this then we had better remember that the player is also a person and also gets a nose. In fact slightly odd things can happen if we combine this with changing the identity of the player. This works:

```
Cleopatra is a woman in Alexandria. The player is Cleopatra.
A nose is a kind of thing. A nose is part of every person.
```

but if those lines are in reverse order then Cleopatra's nose is assembled before she becomes the player, with the result that it ends up called "Cleopatra's nose" rather than "your nose" in play - which is very regal but probably not what we want. To avoid this, settle the player's identity early on in the source text.

All of the assemblies above make objects. Most make these new objects "part of" existing ones, but as we saw, they can also be "in" or "on" them. In fact, though, assemblies work in much more general ways: they can assemble values of almost any kind, placed in almost any relationship. To make use of that, we need to create a new verb, a topic which won't be covered properly until a later chapter, but here goes:

```
A colour is a kind of value. The colours are red, green and blue.

Liking relates various people to various colours. The verb to like
means the liking relation.

Every person likes a colour.
```

Now every time a person is created, so is a colour which that person will like. If there are two people in the world, the player and Daphne, then we now have five colours: red, green, blue, Daphne's colour and the player's colour. Alternatively, we can assemble the other way around:

```
A person likes every colour.
```

Now we're telling Inform that every time a colour is made, a new person is also made - someone who will like that colour. So this sentence effectively makes three new people, one who likes red, one who likes green, and one who likes blue.

Start of Chapter 4: Kinds

Back to §4.14. Duplicates

Onward to §4.16. Names made in assembly

Example 56: ★ **Being Prepared** A kind for jackets, which always includes a container called a pocket.

Example 57: ★★ **Model Shop** An "on/off button" which controls whatever device it is part of.

Example 58: ★★★ **U-Stor-It** A "chest" kind which consists of a container which has a lid as a supporter.

Example 59: ★★★ **The Night Before** Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

---

§4.16. Names made in assembly

Something skated over in the previous section is the question of how Inform gives names to objects (or other values) it creates in an assembly. The standard thing naming combines the names of what's being assembled. For example:

```
A nose is a kind of thing. A nose is part of every person. Antony and
Cleopatra are people.
```

might result in the creation of "Antony's nose", part of Antony, and "Cleopatra's nose", part of Cleopatra. In this way, Inform names the noses after their owners. It will always do this unless there are multiple indistinguishable things being created, as in the "five silver coins are in every banking room" example: those will all just be called "silver coin".

A small pitfall of this is that if we write:

```
Marcus Tullius Cicero is a person.
```

then although "Marcus Tullius Cicero's nose" and "Cicero's nose" are both valid names for the consular nose, "Marcus's nose" is not.

The standard naming scheme is often about right, but as usual Inform offers a way to improve it in particular cases. For example, if we write:

```
Every room contains a vehicle (called its buggy).
```

then we will find the world full of, say, the Garden buggy, the Patio buggy and so on - instead of the Garden vehicle, the Patio vehicle and so on, which is what we would have had without the "called..." part. Similarly, we could write:

```
A person (called its fan) likes every colour.
Every person likes a colour (called his favourite colour).
```

The former would produce new people with names like "Green's fan", whereas the latter would produce new colours with names like "Daphne's favourite colour".

So much for an informal description. Here is exactly what Inform does:

```
(1a) If there is a "called..." text, Inform uses it, expanding out "its" (or
"his" or "her" or "their") to a possessive form of the name of the
owner, so to speak, and "it" (or "he" or "she" or "they" or "him" or
"them") to the name itself.
```

```
(1b) If there's no "called..." text, Inform behaves as if we had written
"(called its K)", where K is the name of the kind.
```

```
(2) If this results in a value which isn't an object being given a name
which already exists, Inform tacks on a number to force the new
name to be different from existing ones: e.g., "Daphne's colour 2",
"Daphne's colour 3", ...
```

(The reason that (2) doesn't affect objects is that objects are allowed to have names clashing with other objects, or no name at all, whereas other values have to have names belonging to themselves alone.)

Start of Chapter 4: Kinds

Back to §4.15. Assemblies and body parts

Onward to §4.17. Postscript on simulation

§4.17. Postscript on simulation

That concludes our tour through the design of the initial state of a simulated world. We have seen how to create rooms and to stock them with containers, supporters, devices, doors, men and women. The player of such a simulation can explore, move things around, open and close certain doors and containers, lock or unlock them provided a suitable key is found, switch machines on or off, and so on.

But that is about all. There is as yet no element of surprise, no aim or sense of progress to be earned, and no narrative thread. We have painted the backcloth, and laid out the properties, but the actors have yet to take the stage.

Start of Chapter 4: Kinds

Back to §4.16. Names made in assembly

Onward to Chapter 5: Text: §5.1. Text with substitutions

Examples from Chapter 4: Kinds

Start of this chapter

Chapter 5: Text

Indexes of the examples

43 **ExampleVouvray** *RB*

Adding synonyms to an entire kind of thing.

The Understanding chapter lays out ways to change how the player can refer to objects, but we may not want to wait that long for some of the basic features. Here, for instance, is how to add synonyms that the player can use to refer to an entire kind of object:

"Vouvray"

```
The Wine Emporium is a room. "Set aside, you rather
suspect, for tourists: this chamber is barrel-vaulted
stone, lined on each side with casks of aging wine.
Discarded brochures here and there advertise Wine
Tours of the Loire Valley in three different languages,
none of them French."

A cask is a kind of thing. A cask is always fixed in place.
Understand "cask" or "barrel" as a cask. Understand
"casks" or "barrels" as the plural of cask.

The Vouvray cask and the Muscadet cask are casks in
the Wine Emporium.

Test me with "get barrels / get barrel / muscadet / x
casks / x muscadet cask".
```

ExampleOdin *RB*

Replacing "You see nothing special..." with a different default message for looking at something nondescript.

In recent years there has been a strong trend towards providing unique descriptions for all implemented objects. Often this is a good idea, but there are also contexts in which we may want to discourage the player from looking too closely at some things and concentrate his attention on just a few interesting ones.

The trick here is that leaving items completely undescribed leads to rather dull exchanges like this:

```
>x table
You see nothing special about the table.
```

...which can leave the player with the impression that the author was simply too lazy to describe everything. So it can be a good idea to replace that default message with a different one more appropriate to the

game. For instance:

```
"Odin"

The House of a Mortal Farmer is a room. "Having two
separate rooms, this house testifies to considerable
wealth and success at agriculture."

The Bedroom is inside from the House.

A chair is a kind of supporter. A chair is always
enterable.

In the House are a table, two chairs, a basket, and a
hearth. On the table is a loaf of bread.

The description of a thing is usually "You give [the noun]
a glance, but it is plainly beneath your attention."
```

Because the description is attached to a whole kind ("thing"), it is really a blanket instruction about many objects at once. More specific instructions always override less specific ones, so we can easily make exceptions. For instance, the following will work correctly:

```
The infant is a man in the basket. The description of the
infant is "So strong and fat that you wonder whether one
of your fellow gods is acquainted with the mistress of
the house-- but it's no concern of yours, of course."

Test me with "x table / x chair / x infant".
```

ExampleSomething Narsty *RB*

A staircase always open and never openable.

In a game with many staircases, we might want:

```
A staircase is a kind of door. A staircase is always open.
A staircase is never openable.
```

Defining the staircase this way means that we will never be able to get away with (for instance) a folding ladder into the attic which is sometimes closed up. So alternatively we might do

```
"Something Narsty"

A staircase is a kind of door. A staircase is usually open.
A staircase is seldom openable.
```

We could then write a rule so that whenever the player types CLIMB [any staircase], the command is diverted to an enter command, while all other doors refuse to respond to such treatment. Still, this kind is now usable, as we see in this trivial example:

```
The ladder is a staircase. It is above the Woodshed and
below the Scary Loft.

Test me with "up".
```

But these are refinements for a later chapter.

ExampleGet Me to the Church on Time

Using kinds of clothing to prevent the player from wearing several pairs of trousers at the

Inform's default handling of wearable things does not make any rules about what can be worn together. Suppose, however, we have a game in which there are a large number of different garments, and we want to keep the player from wearing (say) more than one pair of pants at once:

```
"Get Me to the Church on Time"

A pair of pants and a shirt are kinds of thing. A pair of pants and a shirt are usually wearable.

Some golf pants are a pair of pants. The description is "Checked in red and green, with tiny frolicking gophers every few inches."

Some tuxedo trousers are a pair of pants. The description is "Black, pressed, and slimming."

The frilly shirt is a shirt. The description of the frilly shirt is "She insisted."

The polo shirt is a shirt. The description is "Turquoise and bright yellow, the colors selected by your golfing buddies."

The player wears the golf pants and the polo shirt. The player carries the tuxedo trousers and the frilly shirt.
```

> The Wedding Chapel Dressing Room is a room. "The bride's dressing room is a lavish suite with closets, hangers, dressmaker's dummies, boxes of straight pins and sewing notions, combs, lotions, brushes, and hair fixatives, plus room for fifteen female attendants and a photographer. Before they shoved you out of the room you even got a glimpse of a small reference library including '1001 French Braids' and 'Corset-Lacing For Beginners.'
>
> This is the groom's dressing room. You get a framed photograph of Elvis, a dusty mirror, and the floor space of an average toilet stall."
>
> The dusty mirror and the photograph of Elvis are scenery in the Dressing Room. The description of the mirror is "You can't really get more than a silhouette impression of yourself." The description of Elvis is "He reminds you that you'd better get out there before the organist switches to Hound Dog."

And now the rule itself, borrowed from a later chapter:

> Instead of wearing a pair of pants when the player is wearing a pair of pants (called the wrong trousers):
> say "You'll have to take off [the wrong trousers] first."
>
> Instead of wearing a shirt when the player is wearing a shirt (called the wrong top):
> say "You'll have to take off [the wrong top] first."
>
> When play begins:
> say "From the other side of the door, you hear the organist move on from his instrumental interpretation of 'I Wanna Hold Your Hand' to a somewhat more spirited rendition of 'Help! I Need Somebody!'. Okay, okay, but you've been rushing things along since the 16th fairway, and you can't be more than a half-hour late... Surely that mother of hers can't blame you for that?"

```
Test me with "i / x trousers / wear trousers / x golf pants
/ take off golf pants / wear trousers / x frilly shirt / x polo
shirt / wear frilly shirt / doff polo shirt / wear frilly shirt".
```

If we wanted to, we could make similar kinds for hats, shoes, and so on, and have a simple but effective system of clothing. A more complicated treatment might keep track of layering and describe the player's outfit differently depending on which clothes were outermost -- an example for a later chapter.

---

47 **Example** **Change of Basis** *RB*

Implementing sleeping and wakeful states.

Suppose we want to allow the player to go to sleep some of the time:

```
"Change of Basis"
```

```
A person is either awake or asleep. A person is usually
awake.
```

The important thing to note here is that it does not work to say "the player is either asleep or awake". This is because the player is not necessarily one specific person or thing during the game: the identity of the player can be changed, as we will see later.

So if we want to make rules about the properties of the player, we should attach these rules to the "person" kind.

```
Linear Algebra Class is a room. "The blackboard is
covered with square arrangements of numbers. These
are supposed to convey something to you, but mostly
you're finding them soporific."
```

Now a few rules about changing from one state to the other:

```
Instead of sleeping: now the player is asleep; say "You
drop off."

Instead of doing something other than waking up,
waiting or sleeping when the player is asleep:
    say "Ssh! You're sleeping!"

Instead of sleeping when the player is asleep:
    say "Zzzz."

Instead of waking up when the player is asleep:
    now the player is awake;
    say "You come to suddenly, wiping drool from your
lips."

Instead of doing something other than looking or
sleeping when the player is awake:
    say "You'd really rather just sleep through this."

Test me with "wake up / sleep / look / z / sleep / wake
up / look".
```

ExampleWould you...? *RB*

Adding new properties to objects, and checking for their presence.

For instance, if we want to give some objects a flavor:

```
"Would you...?"
```

```
The House is a room. The mouse is an animal in the
House.
```

```
The player carries some green eggs and a ham.
```

```
A food is a kind of thing that is edible. Food has some
text called flavor. The flavor of food is usually
"Tolerable."
```

Things are, in general, not edible by default, so we have to make them edible specifically in order to allow them to be eaten by the player. Here we've defined food to be edible by default, and we have given it a standard piece of flavor text.

```
The ham and the green eggs are food. The flavor of the
green eggs is "Delicious!"
```

```
After eating something:
    if the noun provides the property flavor, say "[the
flavor of the noun][paragraph break]";
    otherwise say "It's [noun]-flavored."
```

Note that we use "if the noun provides a flavor..." to make sure that the property exists before attempting to use it. Otherwise, there is the risk that we will try to print a property that does not exist, resulting in errors in the game.

We will only get the "It's [noun]-flavored." response if we successfully eat something that is not a food and does not have flavor text. To test this feature, let's suppose something that isn't exactly food but can theoretically be chewed on:

```
The player carries some paper. The paper is edible.
```

Test me with "eat ham / eat green eggs / eat paper".

---

49. **Example** **Straw Boater** *RB*

Using text properties that apply only to some things and are not defined for others.

Sometimes we like to give properties to kinds of thing, but not fill them in in all cases. For instance, we might have vehicles that optionally make noise, and those might have a "movement sound".

All properties have a default value, which we can find by looking in the Kinds tab of the index. This is what the property will be set to automatically, if we do not change it ourselves. In the case of a text property, that is ""; so for instance we might use our movement sound thus:

"Straw Boater"

Boathouse is a room. "A boathouse circa 1915, which -- though in poor repair -- still suggests Sunday afternoon jaunts taken by women in white gowns and men in straw hats."

North of the Boathouse is the Shallow Water. The description of Shallow Water is "Just south is the boathouse, and beyond it are trees and the marble terrace of the house above. The water deepens to the north."

North of Shallow Water is Deep Water. The description of Deep Water is "From here the boathouse has dwindled invisibly to the south, and you have a broad panorama of the shoreline, all the way down to the Skeleton Point Lighthouse in the southeast."

```
A vehicle has some text called the movement sound.
The sailboat and the motorboat are vehicles in the
Boathouse. The movement sound of the motorboat is
"VRRRROOOMMMM..." Understand "boat" as the
sailboat. Understand "boat" as the motorboat.
```

Note that we haven't given the sailboat any movement sound at all.

```
After going somewhere by a vehicle (called cart):
    if the movement sound of the cart is not "", say "[the
movement sound of the cart][paragraph break]";
    continue the action.
```

```
Instead of exiting when the player is in a vehicle and the
location is not the Boathouse:
    say "You're not dressed for a swim."
```

```
Instead of going somewhere when the player is not in a
vehicle:
    say "You'd rather not try to make this journey by
swimming alone."
```

```
Test me with "n / get in sailboat / n / get out / s / get in
motorboat / n / n".
```

Example **The Undertomb 1** *RB*

A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

This example creates a kind of room, "dead end", and gives each a textual description called its "river sound". Note the use of a text substitution to fill in the appropriate river sound text in each dead end.

```
"The Undertomb"
```

A dead end is a kind of room with printed name "Dead End" and description "This is a dead end. You'll have to go back the way you came, consoled only by [river sound]." A dead end is usually dark.

The Undertomb is a dark room. East is a dead end. South is a dead end with printed name "Collapsed Dead End". Northwest is a dead end called the Tortuous Alcove. In the Undertomb is the lantern. It is lit.

A dead end has some text called river sound. The river sound of a dead end is usually "a faint whispering of running water". The Tortuous Alcove has river sound "a gurgle of running water".

Test me with "get lantern / e / w / s / n / nw / se".

---

51 **Example** **The Undertomb 2** *RB*

Flickering lantern-light effects added to the Undertomb.

"Undertomb 2"

Section 1 - Procedure

Brightness is a kind of value. The brightnesses are guttering, weak, radiant and blazing.

A brightness can be adequate or inadequate. A brightness is usually adequate. Guttering is inadequate.

Temperature is a kind of value. 100C specifies a temperature.

A brightness has a temperature. The temperature of a brightness is usually 700C. The temperature of blazing is 1400C. The temperature of radiant is 1100C.

A dead end is a kind of room with printed name "Dead End" and description "This is a dead end, where crags in the uneven rock are caught by the [brightness of the lantern] flame you hold aloft. Despite [river sound] there is no sign of the stream." A dead end is usually dark.

A dead end has some text called river sound. The river sound of a dead end is usually "a faint whispering of running water".

Section 2 - Scenario

The Undertomb is a dark room. "From this dim cross-groined room, exits depart east, south, and northwest."

East is a dead end. South is a dead end with printed name "Collapsed Dead End".

Northwest is a dead end called the Tortuous Alcove. The Tortuous Alcove has river sound "a gurgle of running water".

The player carries a book. The description of the book is "[if the brightness of the lantern is adequate]Many secrets are now yours.[otherwise]No, the print's too tiny by this awful light."

In the Undertomb is a lantern. It is lit. The lantern has a brightness. The lantern is blazing. The description of the lantern is "The lantern shines with a flame at [temperature of the brightness of the lantern]."

After waiting in the Tortuous Alcove when the brightness of the lantern is not guttering:
    now the lantern is the brightness before the brightness of the lantern;
    say "You wait so long that your lantern dims a bit."

Test me with "east / west / get lantern / east / west / south / north / northwest / read book / x lantern / z / x lantern / read book / look / z / x lantern / read book / look / z / x lantern / read book / look".

ExampleThe Crane's Leg 1 *RB*

A description text that automatically highlights the ways in which the object differs from a standard member of its kind.

When you see a long object, you don't have to think that it is too long if being long is the property given by the nature. It is proved by the fact that a duckling, having short legs, will cry if you try to draw them out by force, and that a crane, having long legs, will protest you with tears if you try to cut them with a knife. -- Zhuangzi

Suppose we have an extremely detailed world model in which every object is characterized by many features -- in this example, material and

height, though one could add more. Suppose further that we would like to generate descriptions of these things automatically for the most part, drawing the player's attention only to those aspects of the object that are particularly interesting.

```
"The Crane's Leg"
```

```
Material is a kind of value. The materials are wood,
glass, stone, cloth, paper, clay, and metal. A thing has a
material.
```

```
A height is a kind of value. 3 feet 11 inches specifies a
height. A thing has a height. Definition: a thing is tall if
its height is 6 feet 0 inches or more. Definition: a thing is
short if its height is 2 feet 0 inches or less.
```

So far, we have generally dealt with cases where the property of a thing can be a number (such as 3), a value (such as brightness), or a unit (like height, here). It is also possible for a thing to have a property which names another thing, as in "The mother of the baby trout is the large trout" -- where "mother" is a property, and its value, in the case of the baby trout, is large trout. We would define such a property with a line such as "A fish has a thing called the mother."

In practice, though, this is a bit confusing as syntax; moreover, Inform has a much more powerful construct for talking about the ways in which one object relates to another object. A full discussion of this will have to wait for the chapter on Relations. For now, it is enough to say that we can do this:

```
Imitation relates various things to one thing (called the
ideal). The verb to imitate means the imitation relation.
```

This will allow us to declare that some objects imitate other objects, like so:

```
A chair is a kind of supporter. A chair is always
enterable. A chair is usually wood. The height of a chair
```

```
is usually 4 feet 0 inches. The ordinary chair is a chair.
Every chair imitates the ordinary chair.

A table is a kind of supporter. A table is usually wood.
The height of a table is usually 3 feet 8 inches. The
ordinary table is a table. Every table imitates the
ordinary table.

A rock is a kind of thing. A rock is usually stone. The
ordinary rock is a rock. Every rock imitates the ordinary
rock. The height of a rock is usually 0 feet 3 inches.

A jug is a kind of container. A jug is usually clay. The
ordinary jug is a jug. Every jug imitates the ordinary jug.
The height of a jug is usually 0 feet 8 inches.
```

Now each of these types has one ideal representative which has the fundamental attributes of its kind: the ordinary chair is the most chairlike chair imaginable, the ordinary table is the epitome of tableness, and so on. We are also allowed to refer to "the ideal of the chair", thanks to the way we defined imitation. (Again, the relations chapter offers a much more detailed explanation of how relations may be defined.)

```
The description of a thing is usually "[comparison with
ideal][run paragraph on]".

To say comparison with ideal:
    let the sample be the ideal of the noun;
    if the sample is not a thing:
        say "Perfectly conforming to your expectations of
its type.";
        rule succeeds;
    if the material of the noun is not the material of the
sample:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun], and made of
[material of the noun].";
```

```
            otherwise say "Unusually tall at [height of the
noun], and made of [material of the noun].";
        otherwise:
            say "Distinct mostly in being made of [material of
the noun].";
    otherwise:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun].";
            otherwise say "Unusually tall at [height of the
noun].";
        otherwise:
            say "In every respect [a sample]."



The Pleasure Garden is a room. "At the riverbank, a
pleasing garden, having many curving paths and one
straight."



The low table is a table in the Pleasure Garden. The
height of the low table is 2 feet 3 inches. On the low
table is a metal rock called a gold nugget.



Test me with "x table / x nugget".
```

So far the effect is not very deep, but we could take the auto-description a great deal further: providing a larger and more interesting set of variations; or writing a complicated set of rules such that the player only notices height variations when carrying a ruler; or switching between several player-characters, each of whom notices a different subset of characteristics. But these refinements would require more input from later chapters.

Example **Signs and Portents** *RB*

Signpost that points to various destinations, depending on how the player has turned it.

```
"Signs and Portents"

Seven Dials is a room. The description of Seven Dials is
"There is a signpost, on which seven hands swivel and
swing, freely as weathercocks. They make your present
road now London, now Abingdon; now Weston-super-
Mare, or now Hell."

Seven Dials contains a signpost. The signpost is
scenery. Understand "sign" and "post" as the signpost.

Destination is a kind of value. The destinations are
London, Abingdon, Luton, Weston-super-Mare,
Runnymede, Hell, and Low Noon.

The signpost has a destination.
```

In order to interact with the signpost, we will need to make use of some action rules:

```
Instead of examining the signpost:
    say "[The signpost] currently puts you on the road to
[italic type][the destination of the signpost][roman type],
but it swiftly alters again.";
    now the destination of the signpost is the destination
after the destination of the signpost.

Instead of turning the signpost:
    now the destination of the signpost is the destination
after the destination of the signpost;
    say "With a hand's touch you turn the signpost to
mark your way for [italic type][the destination of the
signpost][roman type]."
```

```
Instead of going north in Seven Dials when the
destination of the signpost is Hell:
    say "It is a path that goes gently ever down and down
with no stumbling block or any distraction at either side;
there are no bandits and no tolls.";
    end the story.


Instead of going north in Seven Dials when the
destination of the signpost is Low Noon:
    say "A long road whose scenery does not change,
nor anything on the horizon move but the sun. When at
last you come to Noon, she hangs above your head like
a hat.";
    end the story finally.


Test me with "x signpost / n / turn signpost / n / turn
signpost / n / turn signpost / n / turn signpost / n".


Test more with "x signpost / n / turn signpost / n / turn
signpost / n / turn signpost / n / turn signpost / turn
signpost / n".
```

ExampleReal Adventurers Need No Help *RB*

Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

Suppose we have an action called "asking for help" that gives the player some hints on request. We've also made it possible to turn this feature off, if the player would like to discourage himself from using the hints too much. Now we need a value that varies to keep track of whether hints are currently permitted or currently not permitted. So we might write:

```
"Real Adventurers Need No Help"
```

```
A permission is a kind of value. The permissions are
allowed and denied.


Hint usage is a permission that varies. Hint usage is
allowed.
```

And under the right circumstances, we change hint usage to denied:

```
Check asking for help for the first time:
    say "Sometimes the temptation to rely on hints
becomes overwhelming, and you may prefer to turn off
hints now. If you do so, your further requests for
guidance will be unavailing. Turn off hints? >";
    if player consents:
        now hint usage is denied;
        say "[line break]Truly, a real adventurer does not
need hints." instead.
```

Then we can refer back to this value later to decide whether we want to display the hint menu or not:

```
Check asking for help:
    if hint usage is denied, say "You have chosen to
eschew hints in this game. Be strong! Persevere!"
instead.


Asking for help is an action out of world. Understand
"help" or "hint" or "hints" as asking for help.


The Realm of Terribly Unjust Puzzles is a room.


Carry out asking for help:
    say "Fine, since you're weak enough to ask: here is a
complete walkthrough: GET EGG. PEEL EGG. SMELL
EGG. DIVIDE YOLK INTO THREE PORTIONS. GIVE
THE SMALLEST PORTION OF YOLK TO THE
GOLDEN GOOSE. ASK THE GOOSE ABOUT
```

```
WHETHER THE SWAN IS TO BE TRUSTED. GIVE
THE LARGEST PORTION OF YOLK TO THE SWAN.
DANCE CONGA. EAT MEDIUM PORTION. STAND ON
HEAD. WEST."


Test me with "hint".
```

Note that it would probably be kinder to offer the player some intermediate level of help, in the actual event.

Example **Early Childhood** *RB*

55 A child's set of building blocks, which come in three different colours - red, green and blue - but which can be repainted during play.

This would be a one-star example if it were not for the repainting:

```
"Early Childhood 1"


A building block is a kind of thing. A red block, a blue
block and a green block are kinds of building block.


The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.


Test me with "look / get red block".
```

But a kind cannot change during play, so this will not do. Instead, the colour will have to be a property of the block. So we might first try this:

```
"Early Childhood 2"
```

```
Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue.

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.

Test me with "look / get red block".
```

Which is fine, so far as it goes, but the colour property is not at all visible to the player, who simply sees "eleven blocks". We thought of colour as being something outwardly apparent, but Inform does not know this. To achieve a better effect, we will need features from distant chapters. The first is an activity called "printing the name of":

```
"Early Childhood 3"

Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue. Before printing the name of a
block: say "[colour] ". Before printing the plural name of
a block: say "[colour] ".

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.

Test me with "look / get red block".
```

This too, however, is unsatisfactory. The individual blocks are correctly described, but we are unable to distinguish them during play: we cannot type "take a green block", for instance. And because the blocks are indistinguishable in play, they are still massed together as "eleven blocks" in room descriptions. We need to go one step further:

```
"Early Childhood 4"

Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue. Before printing the name of a
block: say "[colour] ". Before printing the plural name of
a block: say "[colour] ". Understand the colour property
as describing a block.

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.
```

And now everything works nicely: the blocks are grouped by colour, and can be referred to by colour, and we can even change the colour of an individual block during play, using a bit of extra trickery from later:

```
Understand "paint [something] [colour]" as painting it.
Painting it is an action applying to one thing and one
colour. Check painting it: if the noun is not a block, say
"Paints are only for blocks." instead. Carry out painting
it: now the colour of the noun is the colour understood.
Report painting it: say "The block is now [the colour of
the noun]."

Test me with "get red block / get blue block / g / i / look /
paint blue block red / i / look / paint me red".
```

Example **Being Prepared** *RB*

A kind for jackets, which always includes a container called a pocket.

```
"Being Prepared"
```

```
A jacket is a kind of thing. A jacket is always wearable.


A pocket is a kind of container. A pocket is part of every
jacket. The carrying capacity of a pocket is always 2.


After examining a jacket:
    let target be a random pocket which is part of the
noun;
    say "[The target] contains [a list of things in the
target]."
```

Now we've created the rules that will govern any specific jackets we might happen to put in our game: each one will always have one pocket, which will be able to contain no more than two things. The description of "a list of things" is text with a list, which we will learn about further in a few sections.

Next we might want to create the environment and an actual example of the jacket kind:

```
Tent is a room. "A dome made of two flexible rods and a
lot of bright green ripstop nylon. It bills itself as a one-
man tent, but you'd call it a two-dwarf tent: there is no
way to arrange yourself on its square floor so that you
can stretch out completely."


The hoodie is a jacket. "Your hoodie is balled up in the
corner." The description of the hoodie is "Both elbows
are stained from yesterday's entrenching project."


The hoodie's pocket contains a Swiss army knife and a
folded map. The hoodie is in the Tent.
```

Notice that, since Inform has created a pocket for the hoodie, we can now refer to it by name in our source, giving it any additional properties we need to define. Here we simply put a few items into it.

```
The player wears a whistle. The description of the
whistle is "To frighten bears."


Test me with "x hoodie / get hoodie / get knife / get map
/ i / put hoodie in pocket / put whistle in pocket / put map
in pocket / put knife in pocket / i".
```

Notice that Inform automatically refuses to put the hoodie into its own pocket: as a default, a container cannot contain something of which it is itself a part.

Example **Model Shop** *RB*

57

An "on/off button" which controls whatever device it is part of.

Suppose we're particularly mechanically-minded and would like a game in which all of our mechanical devices have buttons to turn them on and off.

```
"Model Shop"


An on/off button is a kind of thing.


Instead of pushing an on/off button which is part of a
switched off device (called the machine):
    try switching on the machine.
```

Here we are making a rule about how our hypothetical buttons will interact with the machines to which they belong. Instead of pushing... is a rule that pertains to actions, and we will learn more about these in the chapter on actions. "...which is part of a switched off device" provides a specific circumstance - this is only to apply to buttons that are stuck to a machines that can be turned on or off. "(called the machine)" tells Inform that if it finds such a device, it should thereafter refer to it as "the

machine." (The called syntax is explained further in the chapter on Change.)

A set of three more rules will complete our instructions about using buttons to control devices:

```
Instead of pushing an on/off button which is part of a
switched on device (called the machine):
    try switching off the machine.
```

```
Instead of switching on an on/off button which is part of
a device (called the machine):
    try switching on the machine.
```

```
Instead of switching off an on/off button which is part of
a device (called the machine):
    try switching off the machine.
```

Then we hand out buttons with a free hand:

```
One on/off button is part of every device.
```

```
The Model Shop is a room. A model train is a fixed in
place device in the Model Shop. A toy elephant is a
device in the Model Shop.
```

```
Every turn when the model train is switched on:
    say "The model train circles your feet, blowing small
puffs of steam."
```

```
Every turn when the toy elephant is switched on:
    say "The toy elephant waves its trunk at you."
```

```
Test me with "push model train's button / push
elephant's button / g / switch off model train's button".
```

And now the game will have a model train's button and a toy elephant's button.

It may be that we want (as an added nuance) to add other names for these items. While we would want an assembly to create objects such as "Lucy's hand" and not "Lucy hand", it is entirely reasonable to want to talk about the model train button or the elephant button. We could define these additional names like so:

```
Understand "elephant button" or "button on elephant" as
the elephant's button.
```

```
Understand "model train" or "model" or "train" as
"[train]". Understand "[train] button" or "button on [train]"
as the model train's button.
```

In the second case, we are defining [train] to mean any of the three phrases "train", "model", and "model train"; so "[train] button" will match "model train button" or "train button" or "model button" equally well. See the chapter on Understanding for more on how to create alternative phrasings for the player to use.

---

58. **Example** **U-Stor-It** *RB*

A "chest" kind which consists of a container which has a lid as a supporter.

Suppose we want to write a game in which there are a number of chests. Each of these chests will be a container, but have a lid which is a supporter.

```
"U-Stor-It"
```

```
Section 1 - Assemblies and Supporters
```

```
A chest is a kind of container. A chest is always
openable. A chest is usually fixed in place. A chest is
usually closed. The specification of a chest is
"Represents a container with a separately implemented
lid; the lid is itself a supporter."


A lid is a kind of supporter. A lid is part of every chest.
The specification of a lid is "A supporter attached to a
chest, which can only support things when the chest is
closed."
```

(The "specification" of a kind is not really a property, and is used instead to describe the kind in the Index. So the text of these specifications is never found in the game.) Of course, this doesn't get us very far. We will also want the game to correctly interpret variations on "open the chest" and "close the lid", redirecting actions appropriately.

```
Section 2 - Opening and closing


Before opening a lid which is part of a chest (called the
item):
    try opening the item instead.


Before closing a lid which is part of a chest (called the
item):
    try closing the item instead.


Before opening a chest (called the box) when something
is on a lid (called the obstruction) which is part of the
box:
    repeat with item running through things on the
obstruction:
        say "(first removing [the item])";
        try taking the item.
```

```
Instead of opening a chest when something is on a lid
(called the item) which is part of the noun:
    say "You'd have to remove [the list of things on the
item] from the lid first." instead.
```

```
Instead of looking under a lid which is part of a chest
(called the item):
    try opening the item.
```

We may also want to be able to deal with "put in" and "put on" appropriately, even if the player names the wrong part of the object:

```
Section 3 - Insertion and Support
```

```
Before inserting something into a lid which is part of a
chest (called the item):
    try inserting the noun into the item instead.
```

```
Before putting something on a chest when a lid (called
the item) is part of the second noun:
    try putting the noun on the item instead.
```

Furthermore, we don't want the player to be able to put things on the lid while the chest is open:

```
Before putting something on a lid which is part of an
open chest (called the item):
    say "(first closing [the item])";
    try closing the item.
```

```
Instead of putting something on a lid which is part of an
open chest (called the item):
    say "[The item] would need to be closed first."
```

And then we may also want a couple of rules for describing our assembled object nicely:

```
Section 4 - Description in Rooms

Instead of examining a closed chest when something is
on a lid (called the top) which is part of the noun:
    say "[The noun] is closed, and there [is-are a list of
things on the top] on top."

After printing the name of a chest (called the item) while
listing contents of a room:
    if a lid (called the second item) which supports
something is part of the item:
        say " (on which [is-are a list of things on the
second item])";
        omit contents in listing.
```

Now we are free to create entire treasure rooms at a single blow:

```
Section 5 - U-Stor-It Facility

The U-Stor-It Facility is a room. The sea trunk, the
shipping crate, and a metal box are chests in the U-
Stor-It Facility. The metal box contains a sapphire, a
gold coin, and a signed photograph of Babe Ruth.
```

Even though we have never explicitly defined it, the metal box has a "metal box's lid", which we can use at need.

```
The metal box's lid supports a small card. The
description of the small card is "It reads, 'Back in 5 mins
- Pandora.'"

Test me with "open trunk / x card / open metal box / put
all in metal box / get card / put card on box".
```

Example: The Night Before

RB

Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

Suppose that we're going to give every person in the game a nose, but we want references to a nose always to mean the nose of someone *else*, if the player is with one other person. Moreover, on some occasions we're going to be in sight of Rudolph, so actions directed at an unspecified nose should always prefer his.

This relies on a somewhat advanced technique from the Understanding chapter, but since it may become useful with assemblies and body parts, it is worth mentioning here.

```
"The Night Before"

The North Pole is a room. "Here it is: the famous Pole.
From here you can go south (or south-south, or south-
south-by-south); or, alternatively, take refuge inside a
red-and-white-striped cabin." The cabin is scenery in the
North Pole. Instead of entering the cabin, try going
inside.

Santa is a man in the North Pole. "Santa is pacing
around in the snow and trying to psych himself up for
the big night."

Inside from North Pole is the Candy Cane Cabin. The
description of the Cabin is "Striped red and white, but
nothing can make this place seem warm and inviting
since Mrs. Santa ran off with the Tooth Fairy."

The Ice Shelf is south of North Pole. "The ice here has
been smoothed into a kind of runway for easy take-off,
```

```
and ends in a cliff and cold arctic sea." Donner, Vixen,
Blixen, and Rudolph are animals in the Ice Shelf.
```

```
A nose is a kind of thing. A nose is part of every person.
The description of Santa's nose is "It's a bit ruddy. You
don't like to mention it, but Santa's been dipping heavily
into the Grey Goose since Mrs. Santa left town." The
description of a nose is usually "Not terribly exciting."
The description of Rudolph's nose is "See how it glows!"
```

Next, we'll teach Inform some vocabulary to distinguish between the player and everyone else:

```
Definition: a person is other if it is not the player.
```

```
Definition: a thing is selfish if it is part of the player and
the player can see an other person.
```

```
Instead of examining a selfish nose:
    say "You cross your eyes, but can't get a good look."
```

Here is the part that actually determines the preferences. "Does the player mean..." can result in five outcomes: "it is very unlikely", "it is unlikely", "it is possible" (the neutral default), "it is likely", and "it is very likely". This is discussed in greater detail in the Understanding chapter. Here, we want to discourage references to the player's own nose and encourage references to the nose of Rudolph, so:

```
Does the player mean doing something when the noun
is a selfish nose or the second noun is a selfish nose: it
is very unlikely.
```

```
Does the player mean doing something to Rudolph's
nose: it is very likely.
```

And this part is just for decoration:

Rule for writing a paragraph about Rudolph:
    say "The reindeer are already harnessed and waiting impatiently. The brilliance of [Rudolph]'s nose casts an eerie red glow over [the list of unmentioned animals in the location]."

Test me with "x nose / x my nose / x santa's nose / in / x nose / out / s / x my nose / x nose / x rudolph's nose / x donner's nose".

Chapter 5: Text

§5.6. Text with variations; *§5.7. Text with random alternatives*; *§5.8. Line breaks and paragraph breaks*

Contents of *Writing with Inform*
Chapter 4: Kinds
Chapter 6: Descriptions
Indexes of the examples

§5.1. Text with substitutions

In the previous chapter, we gave properties to certain kinds of things in order to change their appearance and behaviour, and saw brief glimpses of one of Inform's most useful devices: text substitution. The following gives a more complete example:

```
"The Undertomb"

A dead end is a kind of room with printed name "Dead End" and
description "This is a dead end. You'll have to go back the way you
came, consoled only by [river sound]." A dead end is usually dark.

The Undertomb is a dark room. East is a dead end. South is a dead
end with printed name "Collapsed Dead End". Northwest is a dead
end called the Tortuous Alcove. In the Undertomb is the lantern. It is
lit.

A dead end has some text called river sound. The river sound of a
dead end is usually "a faint whispering of running water". The
Tortuous Alcove has river sound "a gurgle of running water".
```

The novelty here is the text in square brackets in the first paragraph. They imply more or less what they would when a journalist is quoting something in a newspaper article. The actual words "river sound" are not part of the text. Instead, when Inform prints up

the description of a dead end, it will substitute the appropriate river sound in place of these words.

Thus the description of the Collapsed Dead End is "This is a dead end. You'll have to go back the way you came, consoled only by a faint whispering of running water.", whereas the description of the Tortuous Alcove is "This is a dead end. You'll have to go back the way you came, consoled only by a gurgle of running water." As the player explores these dead ends, subtle differences will appear in their room descriptions.

Start of Chapter 5: Text

Back to Chapter 4: Kinds: §4.17. Postscript on simulation

Onward to §5.2. How Inform reads quoted text

§5.2. How Inform reads quoted text

Text is so fundamental to Inform that the basics had to be covered back in Chapter 2, so let's begin this new chapter with a recap.

Literal text is written in double-quotation marks. It's mostly true that what you see is what you get: the literal text "The Hands of the Silversmith" means just

```
The Hands of the Silversmith
```

But four characters are read in unexpected ways: [,], ' and ". The rules are as follows:

Exception 1. Square brackets [and] are used to describe what Inform should say, but in a non-literal way. For example,

```
"Your watch reads [time of day]."
```

might produce

```
Your watch reads 9:02 AM.
```

These are called "text substitutions". They're highly flexible, and they can take many different forms. But as useful as they are, they do seem to stop us from making actual [and] characters come through on screen. To get around that:

say "[bracket]"

This text substitution expands to a single open square bracket, avoiding the problem that a literal [in text would look to Inform like the opening of a substitution. Example:

```
"He [bracket]Lord Astor[close bracket] would, wouldn't he?"
```

prints as "He [Lord Astor] would, wouldn't he?".

say "[close bracket]"

This text substitution expands to a single close square bracket, avoiding the problem that a literal] in text would look to Inform like the closing of a substitution. Example:

```
"He [bracket]Lord Astor[close bracket] would, wouldn't he?"
```

prints as "He [Lord Astor] would, wouldn't he?".

Exception 2. Single quotation marks at the edges of words are printed as double. So:

```
"Simon says, 'It's far too heavy to lift.'"
```

produces

```
Simon says, "It's far too heavy to lift."
```

This is good because typing a double quotation mark inside the quote wouldn't work - it would end the text then and there. Single quotation marks inside words, such as the one in "it's", remain apostrophes.

The rule looks odd at first, but turns out to be very practical. The only problem arises if we need an apostrophe at the start or end of a word, or a double inside one. Again, substitutions can fix this:

say "[apostrophe/']"

This text substitution expands to a single quotation mark, avoiding Inform's ordinary rule of converting literal single quotation marks to double at the edges of words. Example:

```
Instead of going outside, say "Lucy snaps, 'What's the
matter? You don't trust my cookin[apostrophe] mister?'"
```

produces:

```
Lucy snaps, "What's the matter? You don't trust my cookin'
mister?"
```

A more abbreviated form would be:

```
Instead of going outside, say "Lucy snaps, 'What's the
matter? You don't trust my cookin['] mister?'"
```

which has exactly the same meaning.

say "[quotation mark]"

This text substitution expands to a double quotation mark. Most of the time this is unnecessary because of Inform's rule of converting literal single quotation marks to double at the edges of words, so it's needed only if we want a double-quote in the middle of a word for some reason. Example:

```
"The compass reads 41o21'23[quotation mark]E."
```

which produces: The compass reads 41o21'23"E. (Note that ["] is not allowed; a double-quotation mark is never allowed inside double-quoted text, not even in a text substitution.)

Exception 3. Texts which end with sentence-ending punctuation - full stop, question mark, exclamation mark - are printed with a line break after them. So:

```
say "i don't know how this ends";
say "I know just how this ends!";
```

would come out quite differently - this doesn't affect the appearance of the text, but only the position where the next text will appear. Again, sometimes this is not what we want - the full rules are complicated enough to be worth a whole section later in the chapter.

Start of Chapter 5: Text

Back to §5.1. Text with substitutions

Onward to §5.3. Text which names things

§5.3. Text which names things

We can put almost any description of a value in square brackets in text, and Inform will work out what kind of value it is and print something accordingly. (Only almost any, because we aren't allowed to use commas or more quotation marks inside a square-bracketed substitution.)

say "[(sayable value)**]"**

This text substitution takes the value and produces a textual representation of it. Most kinds of value, and really all of the useful ones, are "sayable" - numbers, times, objects, rules, scenes, and so on. Example:

```
The description of the wrist watch is "The dial reads [time of
day]."
```

Here "time of day" is a value - it's a time that varies, and time is a sayable kind of value, so we might get "The dial reads 11:03 AM."

The values we say most often are objects. If we simply put the name of what we want into square brackets, this will be substituted by the full printed name. We might find:

```
"You admire [lantern]."
= "You admire candle lantern."
```

But this reads oddly - clearly "the" or "a" is missing. So the following substitutions are used very often:

say "[a (object)**]"**

or:
say "[an (object)**]"**

This text substitution produces the name of the object along with its indefinite article. Example:

```
Instead of examining something (called the whatever):
    "You can only just make out [a whatever]."
```

which might produce "You can only just make out a lamp-post.", or "You can only just make out Trevor.", or "You can only just make out some soldiers." The "a" or "an" in the wording is replaced by whatever indefinite article applies, if any.

say "[A (object)**]"**

or:
say "[An (object)**]"**

This text substitution produces the name of the object along with its indefinite article, capitalised. Example:

```
Instead of examining something (called the whatever):
    "[A whatever] can be made out in the mist."
```

which might produce "A lamp-post can be made out in the mist.", or "Trevor can be made out in the mist.", or "Some soldiers can be made out in the mist." The "A" or "An" in the wording is replaced by whatever indefinite article applies, if any.

say "[the (object)**]"**

This text substitution produces the name of the object along with its definite article. Example:

```
Instead of examining something (called the whatever):
    "You can only just make out [the whatever]."
```

which might produce "You can only just make out the lamp-post.", or "You can only just make out Trevor.", or "You can only just make out the soldiers." The "the" in the wording is replaced by whatever definite article applies, if any.

say "[The (object)**]"**

This text substitution produces the name of the object along with its definite article, capitalised. Example:

```
Instead of examining something (called the whatever):
    "[The whatever] may be a trick of the mist."
```

which might produce "The lamp-post may be a trick of the mist.", or "Trevor may be a trick of the mist.", or "The soldiers may be a trick of the mist." The "The" in the wording is replaced by whatever definite article applies, if any.

This may not look very useful, because why not simply put "the", or whatever, into the ordinary text? The answer is that there are times when we do not know in advance which object will be involved. For instance, as we shall later see, there is a special value called "the noun" which is the thing to which the player's current command is applied (thus, if the player typed TAKE BALL, it will be the ball). So:

```
After taking something in the Classroom:
    "You find [a noun]."
```

might produce replies like "You find a solid rubber ball.", "You find an ink-stained blouse.", "You find some elastic bands.", or even "You find Mr Polycarp." (the school's pet hamster, perhaps).

§5.4. Text with numbers

When a numerical value is given in a square-bracketed substitution, it is ordinarily printed out in digits. Thus:

```
"You've been wandering around for [turn count] turns now."
```

might print as "You've been wandering around for 213 turns now.", if the story has been played out for exactly that many commands. But if we prefer:

say "[(number) **in words]"**

This text substitution writes out the number in English text. Example:

```
"You've been wandering around for [turn count in words]
turns now."
```

might produce "You've been wandering around for two hundred and thirteen turns now." The "and" here is natural on one side of the Atlantic but not the other - so with the "Use American dialect." option in place, it disappears.

Either way, though, there is some risk of the following:

```
You've been wandering around for one turns now.
```

We can avoid this using the special substitution:

say "[s]"

This text substitution prints a letter "s" unless the last number printed was 1. Example:

> "You've been wandering around for [turn count in words] turn[s] now."

produces "... for one turn now." or "... for two turns now." as appropriate. Note that it reacts only to numbers, not to other arithmetic values like times (or, for instance, weights from the "Metric Units" extension).

This only solves one case, but it's memorable, and the case is one which turns up often.

Start of Chapter 5: Text

Back to §5.3. Text which names things

Onward to §5.5. Text with lists

Example 60: ★★★ **Ballpark**
A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

---

§5.5. Text with lists

We often want running text to include lists of items.

say "[list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Example:

> "Mr Darcy glares proudly at you. He is wearing [list of things worn by Darcy] and carrying [list of things carried by Darcy]."

And, if this were from a dramatisation of the novel by Miss Fielding rather than Miss Austen, we might find:

> Mr Darcy glares proudly at you. He is wearing a pair of Newcastle United boxer shorts and carrying a self-help book.

If the description matches nothing - for instance, if Darcy has empty hands - then "nothing" is printed.

As with all lists in Inform, the serial comma is only used if the "Use serial comma." option is in force. So by default we would get "a fishing pole, a hook and a sinker", rather than "a fishing pole, a hook, and a sinker".

We then need variations to add indefinite or definite articles, and to capitalise the first item. For example,

> "Mr Darcy impatiently bundles [the list of things carried by Darcy] into your hands and stomps out of the room."

might result in

> Mr Darcy impatiently bundles the self-help book and the Christmas card into your hands and stomps out of the room.

say "[a list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its indefinite article. Example:

> a maritime bill of lading, some hemp rope and Falconer's Naval Dictionary

say "[A list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its indefinite article, and the first is capitalised, so that it can be used at the beginning of a sentence. Example:

> A maritime bill of lading, some hemp rope and Falconer's Naval Dictionary

say "[the list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its definite article. Example:

> the maritime bill of lading, the hemp rope and Falconer's Naval Dictionary

say "[The list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its definite article, and the first is capitalised, so that it can be used at the beginning of a sentence. Example:

> The maritime bill of lading, the hemp rope and Falconer's Naval Dictionary

So much for articles. A more insidious problem comes with something like this:

> "The places you can go are [list of rooms]."

The trouble is that the list may end up either singular or plural. We might be expecting something like:

```
The places you can go are Old Bailey, Bridget's Flat and TV Centre.
```

But if there is only one room, then the result might be:

```
The places you can go are Bridget's Flat.
```

which is wrong. We can get around this with careful wording and a slightly different substitution:

```
"Nearby [is-are list of rooms]."
```

say "[is-are list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. The whole list starts with "is" (if there's one item or none) or "are" (more than one). Examples:

```
is marlin-spike
are maritime bill of lading, hemp rope and Falconer's Naval
Dictionary
```

say "[is-are a list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its indefinite article, and the whole list starts with "is" (if there's one item or none) or "are" (more than one). Examples:

```
is a marlin-spike
are a maritime bill of lading, some hemp rope and
Falconer's Naval Dictionary
```

say "[is-are the list of (description of objects)**]"**

This text substitution produces a list, in sentence form, of everything matching the description. Each item is prefaced by its definite article, and the whole list starts with "is" (if there's one item or none) or "are" (more than one). Examples:

```
is the marlin-spike
are the maritime bill of lading, the hemp rope and Falconer's
Naval Dictionary
```

say "[a list of (description of objects) **including contents]"**

This text substitution produces a list, in sentence form, of everything matching the description, noting any contents in brackets. This is really intended only to be used by the Standard Rules.

Start of Chapter 5: Text

Back to §5.4. Text with numbers

Onward to §5.6. Text with variations

Example 61: ★ **Control Center**
Objects which automatically include a description of their component parts whenever they are examined.

Example 62: ★★ **Tiny Garden**
A lawn made up of several rooms, with part of the description written automatically.

§5.6. Text with variations

Text sometimes needs to take different forms in different circumstances. Perhaps it needs an extra sentence if something has happened, or perhaps only one altered

word.

say "[if (a condition)**]"**

This text substitution produces no text. It's used only for a side-effect: it says that the text following should be said only if the condition is true. That continues until the end of the text, or until an "[end if]" substitution, whichever comes first. If the "[otherwise]" and "[otherwise if]" substitutions are also present, they allow alternatives to be added in case the condition is false. Example:

```
The wine cask is a container. The printed name of the cask
is "[if open]broached, empty cask[otherwise]sealed wine
cask".
```

we find that the cask is described as "a broached, empty cask" when open, and "a sealed wine cask" when closed. A longer example which begins and ends with fixed text, but has two alternatives in the middle:

```
The Customs Wharf is a room. "Amid the bustle of the
quayside, [if the cask is open]many eyes stray to your
broached cask. [otherwise]nobody takes much notice of a
man heaving a cask about. [end if]Sleek gondolas jostle at
the plank pier."
```

say "[unless (a condition)**]"**

This text substitution produces no text. It's used only for a side-effect: it says that the text following should be said only if the condition is false. That continues until the end of the text, or until an "[end if]" substitution, whichever comes first. If the "[otherwise]" and "[otherwise if]" substitutions are also present, they allow alternatives to be added in case the condition is true. Example:

```
The Customs Hall is a room. "With infinite slowness, with
ledgers and quill pens, the clerks ruin their eyesight.[unless
the player is a woman] They barely even glance in your
direction."
```

say "[otherwise]"

or:
say "[else]"

This text substitution produces no text, and can be used only following an "[if ...]" or "[unless ...]" text substitution. It switches from text which appears if the condition is true, to text which appears if it is false. Example:

```
The wine cask is a container. The printed name of the cask
is "[if open]broached, empty cask[otherwise]sealed wine
cask".
```

say "[end if]"

This text substitution produces no text, and can be used only to close off a stretch of varying text which begins with "[if ...]".

say "[end unless]"

This text substitution produces no text, and can be used only to close off a stretch of varying text which begins with "[unless ...]".

say "[otherwise/else **if** (a condition)**]"**

This text substitution produces no text, and can be used only following an "[if ...]" or "[unless ...]" text substitution. It gives an alternative text to use if the first condition didn't apply, but this one does. Example:

```
The wine cask is a container. The printed name of the cask
is "[if open]broached, empty cask[otherwise if
transparent]sealed cask half-full of sloshing
wine[otherwise]sealed wine cask".
```

say "[otherwise/else **unless** (a condition)**]"**

This text substitution produces no text, and can be used only following an "[if ...]" or "[unless ...]" text substitution. It gives an alternative text to use if the first condition didn't apply, and this one is false too.

We sometimes need to be careful about the printing of line breaks:

```
The Cell is a room. "Ah, [if unvisited]the unknown cell. [otherwise]the
usual cell."
```

This room description has two possible forms: "Ah, the unknown cell. ", at first sight, and then "Ah, the usual cell." subsequently. But the second form is rounded off with a line break because the last thing printed is a ".", whereas the first form isn't, because it ended with a space. The right thing would have been:

```
The Cell is a room. "Ah, [if unvisited]the unknown cell.[otherwise]the
usual cell."
```

allowing no space after "unknown cell."

Start of Chapter 5: Text

Back to §5.5. Text with lists

Onward to §5.7. Text with random alternatives

Example 63: ★ **When?** A door whose description says "...leads east" in one place and "...leads west" in the other.

Example 64: ★★★ **Whence?** A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

Example 65: ★★★ **Persephone** Separate the

player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

---

§5.7. Text with random alternatives

Sometimes we would like to provide a little quirky variation in text, especially in messages which will be seen often. We can achieve this with the "[one of]... [or] ... [or] ..." construction.

say "[one of]"

This text substitution produces no text. It's used only for a side-effect: it switches between a number of alternative texts, which follow it and are divided by "[or]" substitutions, according to a strategy given in a closing substitution. Example:

"You flip the coin. [one of]Heads[or]Tails[purely at random]!"

Here there are just two alternatives, and the strategy is "purely at random". Exactly half of the time the text will be printed as "You flip the coin. Heads!"; and the other half, "You flip the coin. Tails!".

say "[or]"

This text substitution produces no text, and can be used only in a "[one of]..." construction. It divides alternative wordings. Example:

"You flip the coin. [one of]Heads[or]Tails[purely at random]!"

There are seven possible endings, each making the choice of which text to follow in a different way:

say "[purely at random]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are chosen uniformly randomly.

say "[then purely at random]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are chosen in sequence until all have been seen, but that after that they are chosen uniformly randomly.

say "[at random]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are chosen at random except that the same choice cannot come up twice running. This is useful to avoid the deadening effect of repeating the exact same message. Example:

```
"The light changes randomly again; now it's [one
of]green[or]amber[or]red[at random]."
```

Here we can safely say the light "changes", because the new colour cannot be the same as the one printed the last time.

say "[then at random]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are chosen in sequence until all have been seen, and then after that, at random except that the same choice cannot come up twice running. Example:

```
"Maybe the murderer is [one of]Colonel
Mustard[or]Professor Plum[or]Cardinal Cerise[then at
random]."
```

say "[sticky random]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that a random choice is made the first time the text is printed, but that it sticks from there on. Example:

```
"The newspaper headline is: [one of]War
Casualties[or]Terrorists[or]Banks[sticky random] [one
of]Continue To Expand[or]Lose Out[sticky random]."
```

Although the newspaper headline will change with each playing, it will not alter during play.

say "[as decreasingly likely outcomes]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are chosen at random, except that the first is most likely to be chosen, the second is next most likely, and so on down to the rarest at the end. Example:

```
"Zorro strides by, [one of]looking purposeful[or]grim-
faced[or]deep in thought[or]suppressing a
yawn[or]scratching his ribs[or]trying to conceal that he has
cut himself shaving[as decreasingly likely outcomes]."
```

There are six outcomes here: the first is six times as likely as the last, and those in between are similarly scaled, so Zorro cuts himself shaving only once in 21 tries, while he looks purposeful almost a third of the time.

But suppose we want to tuck some useful information in these messages, and we want to be sure that the player will see it. Because all of the above options involve randomness, it's possible that an unlucky player might miss a clue placed into only one variant of the message. One fix for this is to make sure that everything turns up sooner or later:

say "[in random order]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. A random order is chosen for the alternative passages of

text, and they are used in that order as the text is printed again and again. When one random cycle finishes, a new one begins. The effect is somewhat like the "shuffle album" feature on an iPod. Example:

> "You dip into the chapter on [one of]freshwater fish[or]hairless mammals[or]extinct birds[or]amphibians such as the black salamander[in random order]."

One small restriction: if there are more than 32 variations, purely random choices will be printed, and there will be no guarantee that repeats are prevented.

Another fix is to avoid randomness altogether:

say "[cycling]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are used one at a time, in turn: after the last one is reached, we start again from the first. Example:

> "The pundits discuss [one of]the weather[or]world events[or]celebrity gossip[cycling]."

say "[stopping]"

This text substitution produces no text, and can be used only to end a "[one of]..." construction. It indicates that the alternatives are used one at a time, in turn: once the last one is reached, it's used forever after. Example:

> "[one of]The phone rings[or]The phone rings a second time[or]The phone rings again[stopping]."

Finally, here's a convenient shorthand for one of the commonest things needed:

say "[first time]"

or:
say "[only]"

This pair of text substitutions causes whatever is between them to be printed only the first time the text is printed. Example:

> "The screen door squeaks loudly as when you open it. [first time]Well, you'll get used to it eventually. [only]"

This is exactly equivalent to

> "The screen door squeaks loudly as when you open it. [one of]Well, you'll get used to it eventually. [or][stopping]";

but easier to read.

Start of Chapter 5: Text

Back to §5.6. Text with variations

Onward to §5.8. Line breaks and paragraph breaks

Example 66: ★ **Radio Daze** A radio that produces a cycle of output using varying text.

Example 67: ★★ **Camp Bethel** Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

---

§5.8. Line breaks and paragraph breaks

Inform controls the flow of text being said so that it will read, to the player, in a natural way. There are two principles:

(a) pieces of text ending with full stop, exclamation or question marks will be followed

by line breaks (or "new lines", as some computer programming languages would call them); and

(b) pieces of text produced by different rules in Inform will be separated by paragraph breaks.

The effect is that authors can forget about paragraph spacing most of the time, but the mechanism is not impossible to fool, so text substitutions are provided to override the usual principles. First, to manipulate line breaks:

say "[line break]"

This text substitution produces a line break. Example:

```
"There is an endless sense of[line break]falling and[line
break]falling."
```

Line breaks are not paragraph breaks, so the result is:

```
There is an endless sense of
falling and
falling.
```

with no extra vertical spacing between these lines.

say "[no line break]"

This text substitution produces no text. It's used only for a side-effect: it prevents a line break where Inform might otherwise assume one. Example:

```
"The chorus sing [one of]Jerusalem[or]Rule, Britannia![no
line break][at random]."
```

Here the "[no line break]" stops Inform from thinking that the exclamation mark means a sentence ending - it's part of the name of the song "Rule, Britannia!". So we get

```
The chorus sing Rule, Britannia!.
```

with no line break between the "!" and ".".

And similarly for paragraph breaks. Because Inform can be pretty trigger-happy with these, the first need is for a way to stop them:

say "[run paragraph on]"

This text substitution produces no text. It's used only for a side-effect: it prevents a paragraph break occurring after the present text is printed, in case Inform might be tempted to place one there. Example:

```
Before taking something, say "Very well. [run paragraph
on]".
```

This allows the reply to, say, TAKE ENVELOPE to be

```
Very well. Taken.
```

rather than

```
Very well.

Taken.
```

which is how texts produced by different rules would normally be shown. (It's a traditional printer's term. See Oldfield's Manual of Typography, 1892, under "When two paragraphs are required to be made into one, or, in technical language, 'to run on'.")

But sometimes we actually want paragraph breaks in unexpected places. One way is to force them outright:

say "[paragraph break]"

This text substitution produces a paragraph break. Example:

> "This is not right.[paragraph break]No, something is terribly wrong."

Paragraph breaks have a little vertical spacing in them, unlike mere line breaks, so the result is:

> This is not right.
>
> No, something is terribly wrong.

More subtly, we can give Inform the option:

say "[conditional paragraph break]"

This text substitution either produces a paragraph break, or no text at all. It marks a place where Inform can put a paragraph break if necessary; in effect it simulates what Inform does every time a "before" or similar rule finishes. If there is text already printed, and text then follows on, a paragraph break is made. But if not, nothing is done. This is sometimes useful when producing a large amount of text which changes with the circumstances so that it is hard to predict in advance whether a paragraph break is needed or not.

Really finicky authors might possibly want to know this:

if a paragraph break is pending:

This condition is true if text has recently been said in such a way that Inform expects to add a paragraph break at the next opportunity (for instance when the present rule ends and another one says something, or when a "[conditional paragraph break]" is made).

Finally, there are two special sorts of paragraph break for special circumstances. They are mainly used by the Standard Rules, and imitate the textual layout styles of traditional IF.

say "[command clarification break]"

This text substitution produces a line break, and then also a paragraph break if the text immediately following is a room description brought about by having gone to to a different room and looking around, in which case a line break should be added. In traditional IF, this is used when clarifying what Inform thinks the player intended by a given command. Example:

```
say "(first opening [the noun])[command clarification break]";
```

might result in

```
(first opening the valise)
You rummage through the valise for tickets, but find nothing.
```

say "[run paragraph on with special look spacing]"

This text substitution produces no text. It's used only for a side-effect: it indicates that the current printing position does not follow a skipped line, and that further material is expected which will run on from the previous paragraph, but that if no further material turns up then a skipped line would be needed before the next command prompt. (It's very likely that only the Standard Rules will ever need this.)

Start of Chapter 5: Text

Back to §5.7. Text with random alternatives

Onward to §5.9. Text with type styles

Example 68: ★★ **Beekeeper's Apprentice** Making the SEARCH command examine all the scenery in the current location.

§5.9. Text with type styles

Inform does not go in for the use of fonts: a work of IF will be rendered with different fonts on different machines anyway, from tiny personal organisers up to huge workstations. However, it does allow for a modest amount of styling.

say "[bold type]"

This text substitution produces no text. It's used only for a side-effect: to make the text following it appear in bold face. "[roman type]" should be used to switch back to normal. Example:

```
"Jane looked down. [bold type]Danger[roman type], the sign
read."
```

say "[italic type]"

This text substitution produces no text. It's used only for a side-effect: to make the text following it appear in italics. "[roman type]" should be used to switch back to normal. Example:

```
"This is [italic type]very suspicious[roman type], said Peter."
```

say "[roman type]"

This text substitution produces no text. It's used only for a side-effect: to return to ordinary Roman type after a previous use of "[bold type]" or "[italic type]".

but there is one other effect we can employ:

say "[fixed letter spacing]"

This text substitution produces no text. It's used only for a side-effect: to make the text following it appear with fixed letter spacing. In variable letter

spacing, a lower case "m" is much wider than an "l", which is natural to the eye since it has been printing practice since the Renaissance. Fixed letter spacing is more like typewriting, and it is best used to reproduce typewritten text or printed notices; it can also be convenient for making simple diagrams. Example:

> "On the door is written: [fixed letter spacing]J45--O-O-O[variable letter spacing]."

say "[variable letter spacing]"

This text substitution produces no text. It's used only for a side-effect: to return to ordinary letter spacing after a previous use of "[fixed letter spacing]".

Whichever effect we use, we should be careful to ensure that we return to normal -- roman type and variable letter spacing -- after any specially-treated text has been printed. Combining these effects (for, say, bold fixed-spaced lettering) is not guaranteed to work, though on some platforms it will.

Start of Chapter 5: Text

Back to §5.8. Line breaks and paragraph breaks

Onward to §5.10. Accented letters

Example 69: ★ **Garibaldi 2** Adding coloured text to the example of door-status readouts.

§5.10. Accented letters

Inform 7 is infused by the English language, so it's a challenge using it to write a work of IF in any other language. (With that said, extensions do exist which have made considerable progress on this problem: nil desperandum.) But even a book in English contains occasional quotations or words borrowed from other tongues, so we are going to need more than plain A to Z.

The world has a bewildering range of letters, accents, diacritics, markers and signs. Inform tries to support the widest range possible, but the works of IF produced by Inform are programs which then have to be run on a (virtual) computer whose abilities

are more constrained: few players will have an Ethiopian font installed, after all. So a degree of caution is called for.

(a) **Definitely safe to use.** Inform's highest level of support is for the letters found on a typical English typewriter keyboard, including both the $ and £ signs (but not the Yen or Euro symbols ¥ and €), and in addition the following:

ä, á, à, ã, å, â and Ä, Á, À, Ã, Å, Â
ë, é, è, ê and Ë, É, È, Ê
ï, í, ì, î and Ï, Í, Ì, Î
ö, ó, ò, õ, ø, ô and Ö, Ó, Ò, Õ, Ø, Ô
ü, ú, ù, û and Ü, Ú, Ù, Û
ÿ, ý and Ý (but not Ÿ)
ñ and Ñ
ç and Ç
æ and Æ (but not œ or Œ)
ß
¡, ¿

These characters can be typed directly into the Source panel, and can be used outside quotation marks: we can call a room the Église, for instance.

(b) **Characters which can safely be used, but will be simplified.** As it reads in the text, Inform silently converts all kinds of dash (en-rules, em-rules, etc.) to simple hyphens; converts the multiplication symbol to a lower case "x"; converts all kinds of space other than tabs (em-spaces, non-breaking spaces, etc.) to simple spaces, and all kinds of quotation marks to "straight" (non-smart) marks.

(c) **Characters which can be used provided they are in quoted text (other than boxed quotations), and which will probably but not certainly be visible to the player**. All other Latin letter-forms, including the œ ligature, East European forms such as ő, ş and ž, and Portuguese forms such as ũ; the Greek and Cyrillic alphabets, with their associated variants and accents; and the principal currency symbols, such as € and ¥. Such characters are not legal in unquoted text: so we could write

```
The Churchyard is a room. The printed name of the Churchyard is
"Łodz Churchyard".
```

but not

```
Łodz Churchyard is a room.
```

Moreover, the player is not allowed to type these characters in commands during play: or rather, they will not be recognised if he does. They are for printing only.

(d) **Characters which might work in quoted text, or might not**. The Arabic and Hebrew alphabets are fairly likely to be available; miscellaneous symbols are sometimes legible to the player, sometimes not. Other alphabets are chancier still. (If a work of IF depends on these being visible, it may be necessary to instruct players to use specific interpreters, or to provide a way for the player to test that all will be well.)

Start of Chapter 5: Text

Back to §5.9. Text with type styles

Onward to §5.11. Unicode characters

§5.11. Unicode characters

As we have seen, Inform allows us to type a wide range of characters into the source text, although the more exotic ones may only appear inside quotation marks. But they become more and more difficult to type as they become more obscure. Inform therefore allows us to describe a letter using a text substitution rather than typing it directly.

Unicode characters can be named (or numbered) directly in text. For example:

```
"[unicode 321]odz Churchyard"
```

produces a Polish slashed L. If the Unicode Character Names or Unicode Full Character Names extensions are included, characters can also be named as well as numbered:

```
"[unicode Latin capital letter L with stroke]odz Churchyard"
```

The Unicode standard assigns character numbers to essentially every marking used in text from any human language: its full range is enormous. (Note that Inform writes these numbers in decimal: many reference charts show them in hexadecimal, or base 16, which can cause confusion.) Inform can only handle codes [unicode 32] up to [unicode 65535], so it is not quite so catholic, but the range is still enormous enough that code numbers are unfamiliar to the eye. Inform therefore allows us to use the official Unicode 4.1 names for characters, instead of their decimal numbers, *provided* we have Included the necessary extension like so:

```
Include Unicode Character Names by Graham Nelson.
```

This extension provides names for some 2900 of the most commonly used characters. It means, for instance, that we can write text such as:

```
"Dr Zarkov unveils the new [unicode Hebrew letter alef] Nought
drive."
"Omar plays 4[unicode black spade suit] with an air of triumph."
```

Admittedly, these can get a little verbose:

```
"[unicode Greek small letter omega with psili and perispomeni and
ypogegrammeni]"
```

But before getting carried away, we should remember the hazards: Inform allows us to type, say, "[unicode Saturn]" (an astrological sign) but it appears only as a black square if the resulting story is played by an interpreter using a font which lacks the relevant sign. For instance, Zoom for OS X uses the Lucida Grande and Apple Symbol fonts by default, and this combination does contain the Saturn sign: but Windows Frotz tends to use the Tahoma font by default, which does not. (Another issue is that the fixed letter spacing font, such as used in the status line, may not contain all the characters that the font of the main text contains.) To write something with truly outré characters is therefore a little chancy: users would have to be told quite carefully what interpreter and font to use to play it.

The "Unicode Character Names" extension, which is pre-installed in the standard distribution of Inform, defines names for the Latin, Greek, Cyrillic, Hebrew and Braille alphabets, together with currency and miscellaneous other symbols, including some for drawing boxes and arrows. It is only optionally installed because even this is quite large: but in case it should still prove inadequate, an alternative can be used:

```
Include Unicode Full Character Names by Graham Nelson.
```

This includes all 12,997 named characters in the 16-bit range of the Unicode 4.1 standard: it is the size of a small novel and its inclusion will slow Inform down. But if you want to experiment with Arabic, ecclesiastical Georgian, Cherokee, Tibetan, Syriac, the International Phonetic Alphabet, hexagrams or the unified Canadian aboriginal syllabics, "Unicode Full Character Names" (again built into Inform) is the extension for you.

Start of Chapter 5: Text
Back to §5.10. Accented letters
Onward to §5.12. Displaying quotations

Example 70: ★★★ **The Über-complète clavier** This example provides a fairly stringent test of exotic lettering.

---

§5.12. Displaying quotations

Text is normally printed in between the typed commands of the player, rolling upwards from the bottom of the screen, as if a dialogue is being typed by an old-fashioned teletype. But it can also be displayed in a bolder way, floating above the main text, and this is sometimes used to display quotations.

display the boxed quotation (text)

This phrase displays the given text on screen in an overlaid box. For reasons to do with the way such quotations are plotted onto the screen, their text is treated literally: no substitutions in square brackets are obeyed. The quotation will only ever appear once, regardless of the number of times the "display the boxed quotation ..." phrase is reached. Rather than being shown immediately - and thus, probably, scrolling away before it can be seen - the display is held back until the next command prompt is shown to the player. Example:

```
After looking in the Wabe, display the boxed quotation
    "And 'the wabe' is the grass-plot round
    a sun-dial, I suppose? said Alice,
    surprised at her own ingenuity.

    Of course it is. It's called 'wabe,'
    you know, because it goes a long way
    before it, and a long way behind it --
```

```
                -- Lewis Carroll".
```

This was the original example used in *Trinity*, by Brian Moriarty, which invented the idea. A player exploring Kensington Gardens comes upon a location enigmatically called The Wabe; and by way of explanation, this quotation pops up.

Note that exotic accented characters, such as the "Ł" in "Łodz", can't be displayed in boxed quotations. This is only a simple feature, and we should go in search of a suitable extension for fancier screen effects if we would like to do more.

Start of Chapter 5: Text

Back to §5.11. Unicode characters

Onward to §5.13. Making new substitutions

§5.13. Making new substitutions

If we have some textual effect which needs to occur in several different messages, we might want to create a new text substitution for it. For instance:

```
The Missile Base is a room. "[security notice]Seems to be a futuristic
missile base." M's Office is east of the Missile Base. "[security
notice]Admiral Sir M.- M.- glares up from his desk."


To say security notice:
    say "This area is a Prohibited Place within the meaning of the
Official Secrets Act 1939. "
```

This is only the tip of the iceberg in how to define ways to do things using "To...", as we shall see. The definition makes "say the security notice" a new phrase known to Inform. A text substitution is exactly a phrase whose name begins with "say" (well - except for the "say" phrase itself), so the effect is that "[security notice]" is a new text substitution. Several of the examples in this chapter make use of this trick.

Inform often ignores the casing of the text it reads, but sometimes uses it as a clue to meaning. We have already seen that "[an item]" and "[An item]" produce different results, for instance. Similarly, it's possible to define two text substitutions which are the same except for the initial casing. We might write:

```
To say Security Notice:
    say "THIS AREA IS A PROHIBITED PLACE WITHIN THE
MEANING OF THE OFFICIAL SECRETS ACT 1939. "
```

And now Inform will act on "[Security Notice]" differently from "[security notice]".

See **The phrasebook** *for other forms of phrase besides To say...*

Start of Chapter 5: Text

Back to §5.12. Displaying quotations

Onward to Chapter 6: Descriptions: §6.1. What are descriptions?

Example 71: **Fifty Ways to Leave Your Larva** Using text substitution to make characters reply differently under the same circumstances.

Example 72: **Fifty Times Fifty Ways** Writing your own rules for how to carry out substitutions.

Examples from Chapter 5: Text

Start of this chapter

Chapter 6: Descriptions

Indexes of the examples

ExampleBallpark *RB*

A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

```
Sometimes it is more sensible to describe numbers roughly than in exact
```

terms. For instance, we might want to have our player perceive "many people" rather than "forty-two people" on entering a room. To achieve this, we might write our own "to say" phrase.

```
"Ballpark"

To say (count - a number) in round numbers:
    repeat through the Table of Numerical Approximation:
        if count is less than threshold entry:
            say "[approximation entry]";
            rule succeeds.
```

Phrases will be explained more thoroughly in a later chapter, but as we have already seen in the examples, we can make a "To say..." phrase that will allow us to create our own text substitutions. In this case, we are going to replace the specific number with a vaguer one chosen from a chart, so:

```
Table of Numerical Approximation

threshold  approximation
1          "no"
2          "one"
3          "a couple of"
6          "a few"
15         "some"
21         "many"
100        "lots and lots of"
```

The idea here is that we will work our way through the table until we hit a line where the threshold number is higher than the number we want to express, and then print that output: so if we have less than one item, we'll print "no"; if we have more than none but less than two, we'll print "one"; if we have less than three, we'll print "a couple of"; if we have three, four, or five (but not six), we'll print "a few."

```
A room has a number called the population. The
population of a room is usually 0. The description of a
room is usually "You observe [population of the location
in round numbers] [if population of the location is
```

```
1]person [otherwise]people [end if]here.".


The Stadium is a room. The Hot Dog Stand is west of
the Stadium. The Women's Restroom is south of the
Stadium.


The population of the Stadium is 500. The population of
the Hot Dog Stand is 3. The population of the Restroom
is 750.


Test me with "w / e / s".
```

---

61 Example Control Center *RB*

Objects which automatically include a description of their component parts whenever they are examined.

It is straightforward to make a rule that anything with parts must mention all those parts during an EXAMINE command:

```
"Control Center"


After examining a thing when something is part of the
noun:
    say "[The noun] includes [a list of things which are
part of the noun]."


The Control Center is a room. "Here you are at the
Control Center of the universe."
```

```
The Universe Management Computer is a fixed in place
thing in the Control Center. "The Universe Management
Computer sits directly before you, unguarded." The
description of the Universe Management Computer is
"The computer is so large that you would be unable to
operate it all from one position. Alas, it does not come
with a manual."


A chartreuse indicator light, an ennui meter, a golden
knob settable to 15,000 positions, a toothpick dispenser,
and a button labeled RESTART are part of the Universe
Management Computer.


The command chair is an enterable supporter in the
Control Center. It is pushable between rooms. "Because
the computer is too large for you to reach all of the front
panel from a standing position, there is a command
chair on casters which allows you to push back and
forth." The description of the command chair is "Quite
ordinary, really, but for the heady rush of power that
comes of sitting in it.". Some casters are part of the
command chair.
```

Now whenever we look at any object with components, we will first see the description, then a list of parts which belong to it. The following refinement brings in elements of later chapters, but it may be worth noting: because we've written our rule as an "After examining...", anything that pre-empts the operation of the examine command will also prevent that rule from occurring. So for instance:

```
A hair-thick needle is part of the ennui meter.


Instead of examining the ennui meter: say "You can't be
bothered."
```

...would not result in the needle being mentioned.

```
Test me with "x chair / x computer / x ennui meter"
```

Example Tiny Garden *RB*

A lawn made up of several rooms, with part of the description written automatically.

Sometimes we want to make a list of something too complicated to express in a say list... phrase. When this happens, we can instead mark all the items we want to mention as "marked for listing".

In this case, we have a lawn area made up of four rooms. We want each room to automatically describe the directions leading to the other parts of the lawn. To do this, we will first determine which directions are relevant and mark those for listing, then list them.

```
"Tiny Garden"

The Herb Garden is a room. "Along this side of the
house run your great-aunt's herb beds."

A Grassy Room is a kind of room. The printed name of
a Grassy Room is usually "Lawn". The description of a
Grassy Room is "The grass underfoot is thick and
green. The lawn extends to [grassy directions] from
here."
```

The following phrase goes through all the directions in the compass and marks the ones that are interesting to us at the moment.

```
To say grassy directions:
    repeat with that way running through directions:
        if the room that way from the location is a grassy
room,
            now that way is marked for listing;
    say "[a list of directions which are marked for listing]";
    now every direction is not marked for listing.
```

Lawn1 is west of the Herb Garden. It contains a picnic table and a wicker basket. Lawn2 is south of Lawn1 and southeast of Lawn4. Lawn3 is southwest of Lawn1, west of Lawn2, and south of Lawn4. Lawn4 is west of Lawn1. Lawn4 contains a birdbath. The birdbath is fixed in place.

Lawn1, Lawn2, Lawn3, and Lawn4 are Grassy Rooms.

Test me with "w / s / w / n".

---

63 **ExampleWhen?** *RB*

A door whose description says "...leads east" in one place and "...leads west" in the other.

Very simple, but quite frequently useful:

"When?"

The temporal vortex is an open door. It is west of Yesterday and east of Today. "A whirling temporal vortex leads [if the player is in Yesterday]west[otherwise]east[end if]."

---

64 **ExampleWhence?** *RB*

A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

It would be fairly tedious reading to have a large game full of doors that describe themselves this way. Nonetheless, if we insisted we could use our knowledge of the map as leverage to make every door in the game describe itself automatically.

To do this, we make use of the phrase "direction of (the door) from (a room)" -- in this case, the direction of the door we're looking at when viewed from the player's location. Thus:

```
"Whence?"

The temporal vortex is an open door. It is west of
Yesterday and east of Today.

The initial appearance of a door is usually "Nearby [an
item described] leads [if the other side of the item
described is visited][direction of the item described from
the location] to [the other side][otherwise][direction of
the item described from the location][end if]."

Test me with "w / e".
```

Yet a further variation on this, which can automatically understand "the east door" and "the west door" when appropriate, may be found in the example "Whither?".

ExamplePersephone *RB*

Separate the player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

If we wanted, we might replace the rule for taking inventory as follows:

```
"Persephone"

Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying
```

```
[a list of things carried by the player][otherwise][We]['re]
empty-handed";
    say "[if the player wears something]. [We]['re] wearing
[a list of things worn by the player][end if]."
```

The Fancy Party is a room. The player carries a sword, a strawberry stem, and 20 credits worth of platinum. The player wears a sash indicating lordhood.

Test me with "i / take off sash / i / drop all / i".

---

66 **Example** **Radio Daze** *RB*

A radio that produces a cycle of output using varying text.

```
"Radio Daze" by Jon Ingold
```

The Living Room is a room. "A long couch, set up so you can see your wireless set. Not that you need to see it, of course."

The long couch is an enterable scenery supporter in the living room. Instead of entering the long couch when the radio was switched off, say "Better turn the radio on before you get comfortable." Report entering the couch: say "You settle yourself down to listen." instead. Instead of listening when the radio is switched on, stop the action.

The radio is a device in the living room. The radio is switched off. "[if switched on]The radio burbles on[otherwise]The radio is off[end if]." Check switching

```
off the radio when the player is on the long couch: say
"You can't reach the radio from here." instead.


Every turn when the radio is switched on:
    say "[one of]Two characters in the radio play have
begun an argument[or]The argument continues[or]The
play continues[stopping]: [one of]'Did not!'[or]'Did
too!'[or]'Did I?'[or]'You did!'[or]'I couldn't have,
Martha!'[or]'But you did, Tom!'[cycling]"


Test me with "sit on couch / turn on radio / sit on couch/
listen / g / g / g / g / g".
```

ExampleCamp Bethel *RB*

Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

```
"Camp Bethel"


Camp Bethel Kitchen is a room.
```

One use for text alternatives is to change the description of a room after first visiting. We've already seen, in the example "Slightly Wrong", how to do this with "[if visited] ... [otherwise] ... [end if]". But since the first description is printed once and the second description on all subsequent occasions, we could equally well write

```
The description of Camp Kitchen is "[one of]You've
never been into the kitchen before, though you've spent
many an hour in the dining lodge. The place is larger
than you would have expected, and it has none of the
fake rustic touches of the rest of the camp[or]A tidy,
efficient industrial kitchen, without any of the kitsch
rusticity found elsewhere[stopping]."
```

We might also want to liven up the behavior of people and animals, who are probably not doing the exact same thing every time we glance in their direction. There are more complex techniques for modeling the behavior of characters, as we will see in the chapters on Advanced Actions and Activities; but if we just want some textual variety, we might write something like:

```
Jeremy is a man in the Camp Bethel Kitchen. "Jeremy
stands at his station, [one of]peeling white
onions[or]briskly dicing onions[or]chopping
celery[or]peeling carrots[or]tying fresh herbs together
with string[or]putting all the vegetables into a large stock
pot[or]watching over his boiling vegetable
stock[cycling]."
```

And since (textual variation or not) we do want the player to be able to see all these objects:

```
Jeremy carries white onions, celery, carrots, and herbs.
Jeremy's station is scenery in the kitchen. It is a
supporter.
```

Jeremy is following a sequence of actions to do an implied task (still somewhat robotically, but it will do for now). Animals might be a bit more capricious, though:

```
Fluffy is an animal in the Camp Bethel Kitchen. "[one
of]Fluffy is chasing its tail[or]Fluffy is staring out the
window[or]Fluffy is rubbing itself against your leg[purely
at random]."
```

```
A housefly is an animal in the Camp Bethel Kitchen. "A
large housefly [one of]lands on a countertop[or]flies
around noisily[or]circles Jeremy's chef hat[at random]."
```

The housefly's description is merely "at random" rather than "purely at random" because we want to show it doing a different thing each turn, whereas Fluffy could plausibly stare out the window for five turns in a row.

There are more complex ways to change and override the initial

descriptions of people and things; if text alternatives do not get us far enough, we can turn to the "rule for writing a paragraph about," documented in the Activities chapter.

Another frequent use of text alternatives is to give characters a bit of variety in things they're likely to say many times in the course of a game:

```
Instead of telling Jeremy about something:
    say "Jeremy looks [one
of]surprised[or]intrigued[or]nonplussed[at random]. '[one
of]You don't say[or]That's very interesting[or]Do go
on[or]I wish I'd known that sooner[at random]!'".
```

Or, with somewhat more complexity:

```
Instead of asking Jeremy about something:
    say "'[one of]Sorry,[or]I'm afraid[or]Hm,[at random]
[one of]I don't know much about that[or]you've got me
there[or]I haven't the faintest[at random],' Jeremy [one
of]drawls[or]replies[or]comments[or]exclaims[at
random]";
    say "[one of][or] huskily[or] throatily[or] silkily[or] in a
deep manly voice[as decreasingly likely outcomes]."
```

Notice that, in that last line, our first option is entirely blank. If we put nothing as an element of the text alternatives list, this means that printing nothing at all is a viable alternative. In fact, we've made this the most common probability out of the decreasingly likely outcomes, so that five times in fifteen, or a third of the time the text is run, there will be no modifier printed at all.

```
Test me with "look / g / g / g / ask Jeremy about his
feelings for me / ask jeremy about his amnesia / tell
Jeremy about my unborn child".
```

As this example (alas) reveals, text alternatives will not go all the way toward making our characters into compelling conversationalists; we will have to wait until we know more about Actions. But at least we have abolished the default responses, and given Jeremy a touch of personality, however witless.

We have to create a suitable action and say what it does, and to repeat what we do through all the scenery items. That needs material from subsequent chapters, but is quite ordinary Inform all the same:

```
"Beekeeper's Apprentice"

Studying the vicinity is an action applying to nothing.

Report studying the vicinity:
    if the location does not contain something which is
scenery:
        say "There's little of interest in the [location]."
instead;
    repeat with point of interest running through scenery
in the location:
        say "[point of interest]: [run paragraph on]";
        try examining the point of interest.

Understand "search" as studying the vicinity.

The Yard is a room.

The hive and the honey are scenery things in the Yard.
The description of the hive is "The honeycombed hive is
all around you, thrumming with life." The description of
the honey is "Wax-sealed honey has been cached in
many of the hexagonal nurseries."
```

```
Test me with "search".
```

The reason for this example is to show the use of saying "[run paragraph on]". It means we have output such as:

```
>search
hive: The honeycombed hive is all around you,
thrumming with life.

honey: Wax-sealed honey has been cached in many of
the hexagonal nurseries.
```

Without the running on, the prompts "hive:" and "honey:" would be separated from the descriptions following them, which would look a little odd.

---

69 **ExampleGaribaldi 2** *RB*

Adding coloured text to the example of door-status readouts.

The extension "Basic Screen Effects" provides a few more type styles, in the form of coloured lettering. The colours available are red, yellow, green, blue, white, magenta, and cyan, as well as the usual black; and to restore the player's default screen colour, we say "default letters".

Thus if we wanted to highlight locked and unlocked doors in our security readout example:

```
"Garibaldi"

Include Basic Screen Effects by Emily Short.

The security readout is a device. The description of the
readout is "The screen is blank."
```

```
Instead of examining the switched on security readout:
    say "The screen reads: [fixed letter spacing]";
    say line break;
    repeat with item running through doors:
        say line break;
        say " [item] ([front side of the item]/[back side of the
item]): [if the item is locked][green
letters]LOCKED[default letters][otherwise][red
letters]UNLOCKED[default letters][end if]";
    say variable letter spacing;
    say paragraph break.



The player carries the security readout.



The Docking Bay is a room. The inner airlock is a door.
It is north of the Docking Bay and south of the Zocalo.
The inner airlock is lockable and unlocked. The outer
airlock is lockable and locked. It is a door. It is south of
the Docking Bay and north of Space.



The quarantine seal is a door. It is west of the Zocalo
and east of Medlab. Quarantine seal is locked.



The security pass unlocks the inner airlock. The player
carries the security pass.



Test me with "x readout / turn on readout / x readout /
lock inner airlock with security pass / x readout".
```

Note that this extension does not currently produce the desired effects when compiling with the Glulx setting; to see it working, make sure that the settings tab is set to compile to the Z-machine.

The following example puts Inform's support for exotic lettering through its paces. It was useful in testing Inform but is not a very instructive read: still, it does provide a test story file for interpreters, so we are including the source here as an example.

```
"The Über-complète clavier"

The story headline is "Pushing the Limits of Unicode in
IF". The story description is "This is a demanding test
for Unicode compliance by Z-machine interpreters."

Include Unicode Character Names by Graham Nelson.

Include Basic Screen Effects by Emily Short.

The Château Bibliothèque Français is east of the
Deutsche Universität Bücherei. "From this Borgesian
construction, doorways lead into anterooms in each of
the four cardinal directions." South of the Bibliothèque is
the Miscellany Mañana. North of the Bibliothèque is the
Íslendingabók. East of the Bibliothèque is Alphabet
Soup.

A framed photograph of Icelandic Prime Minister Halldór
Ásgrímsson, a ruler measuring Ångströms, a Bokmål-
Lëtzebuergesch Lëtzebuergesch-Bokmål dictionary and
a ticket to Tromsø via Østfold are in the Íslendingabók.

A paper by Karl Weierstraß, a general feeling of Ärger,
```

an old Österreich passport and the Bach Clavier-Übung open at the fugue à 4 are in the Bücherei.

The painting of École normale superiéure students singing Ça ira, the frankly lesser-known journal of Niccolò Polo, Così fan tutte on CD, an extract of Herodotus concerning Artaÿctes and the exit sign reading À BIENTÔT are in the Bibliothèque.

A wicker basket marked CHLOË is in the Bibliothèque. A ginger cat is in the basket.

A guide to Æsop for naïve æsthetes, Lönnrot's Kalevala, a creed according to the Bahá'ís, FALARÃO magazine, an Estonian poem by Tõnu Trubetsky, a Portuguese-Italian recipe for macarrão, a stripy hanging CANDY PIÑATA bag, a ¿¡Punctuation Turned Upside Down¿¡ pamphlet, an Italian brewers' anti-violence poster declaring BÓTTE NON BÒTTE, a map of È and a dusty book titled The Parnasum of Luís Vaz bearing CAMÕES on its spine are in Miscellany Mañana.

The description of the map is "È is a province in the People's Republic of China."

In Mañana is something called ÂÊÎÔÛ - The Official Journal of the Society for Vowels bearing Circumflexes.

In Mañana is something called âêîôû comic - the youth edition.

The description of Alphabet Soup is "A bewildering place of glyphs, sigils and signs. The Library proper leads back west: steps lead upwards to an Observatory,

or downwards into what seems to be a dangerous area. A gaming lounge lies to the south."

The Greek Alphabet, the Cyrillic Alphabet, the Hebrew alphabet, and the embossed plaque are in Alphabet Soup. The description of the Greek alphabet is "αβγδεζηθικλμνξοπρςστυφχψω.". The description of the Hebrew alphabet is "אבגדהוזחטיךכלמנסעף פץקרשת.". The description of the Cyrillic alphabet is "абвгдежзийклмнопрстуфхцчшщъыьэюя.".

Instead of examining the plaque:
    say "It seems to be a sign in Braille: ";
    say unicode Braille pattern dots-24, " (I), ",
        unicode Braille pattern dots-1345, " (N), ",
        unicode Braille pattern dots-124, " (F), ",
        unicode Braille pattern dots-135, " (O), ",
        unicode Braille pattern dots-1235, " (R), ",
        unicode Braille pattern dots-134, " (M)."

The Gaming Lounge is south of Alphabet Soup. The chess position and the book of puzzle canons are in the Gaming Lounge.

The Georges de la Tour painting Le Tricheur is in the Gaming Lounge. "Hanging on one wall is Georges de la Tour's masterpiece Le Tricheur (the card-sharp). Visible are 8[unicode black diamond suit], 9[unicode black diamond suit], A[unicode black diamond suit], A[unicode black spade suit], 6[unicode black club suit] but not one of them has a [unicode black heart suit]."

The description of Le Tricheur is "If they'd been dice-players instead, they might have thrown [unicode die face-1], [unicode die face-2], [unicode die face-3], [unicode die face-4], [unicode die face-5] or [unicode die face-6], but as it is they stick to cards."

The description of the book of canons is "A typical fugue is no. 13 (Tovey: [unicode eighth note] = 110) in F[unicode music sharp sign] minor, but you can also make out keys like A[unicode music flat sign] and G[unicode music natural sign]."

The empty square text is text that varies. To say empty: say the empty square text.

To display the board:
    say empty, empty, empty, empty, empty, empty, unicode black chess king, empty, line break;
    say empty, empty, empty, unicode black chess queen, empty, empty, unicode black chess pawn, empty, line break;
    say unicode black chess pawn, empty, empty, unicode black chess bishop, unicode black chess pawn, empty, empty, unicode black chess pawn, line break;
    say empty, empty, empty, unicode black chess pawn, empty, unicode black chess rook, empty, empty, line break;
    say empty, unicode black chess pawn, empty, unicode white chess pawn, unicode black chess pawn, empty, empty, empty, line break;
    say empty, empty, empty, unicode black chess bishop, unicode white chess queen, empty, unicode white chess pawn, unicode white chess pawn, line break;
    say unicode white chess pawn, unicode white chess pawn, empty, unicode white chess bishop, empty, unicode black chess rook, unicode white chess bishop, empty, line break;
    say empty, unicode white chess knight, empty, empty, unicode white chess rook, empty, unicode white chess rook, unicode white chess king, line break.

Instead of examining the chess position:
    say "Fritz Saemisch - Aron Nimzowitsch, Copenhagen 1923: the Immortal Zugzwang Game. Nimzowitsch (black), observing that white will very soon have to play a terrible move, has just advanced his h pawn for no reason other than to wait. So it is white to play...";
    say "[fixed letter spacing].....k. [line break]...q..p. [line

break]p..bp..p [line break]...p.r.. [line break].p.Pp... [line break]...bQ.PP [line break]PP.B.rB. [line break].N..R.RK [variable letter spacing][line break]";
    say "'White must, willy-nilly, eventually throw himself upon the sword', in Nimzowitsch's commentary. ";
    say "We will now try to display the same position using chess-piece symbols in a Unicode font.";
    say fixed letter spacing;
    now the empty square text is " ";
    display the board;
    say variable letter spacing.

The Astrological Observatory is above Alphabet Soup.

The planets are in the Observatory. "Diagrams of the planets are scattered across the dome: Sun [unicode Sun], Mercury [unicode Mercury], Venus [unicode Female Sign], Earth [unicode Earth], Moon [unicode First Quarter Moon] and [unicode Last Quarter Moon], Mars [unicode Male Sign], Jupiter [unicode Jupiter], Saturn [unicode Saturn], Uranus [unicode Uranus], Neptune [unicode Neptune], Pluto [unicode Pluto] and one or two comets [unicode Comet]. Fainter, but all around, you see stars black [unicode black star] and white [unicode white star]."

The constellations are in the Observatory. "Ringing the dome are the constellations: Aries [unicode Aries], Taurus [unicode Taurus], Gemini [unicode Gemini], Cancer [unicode Cancer], Leo [unicode Leo], Virgo [unicode Virgo], Libra [unicode Libra], Scorpius [unicode Scorpius], Sagittarius [unicode Sagittarius], Capricorn [unicode Capricorn], Aquarius [unicode Aquarius], Pisces [unicode Pisces]."

The weather almanac is in the Observatory. The description of the almanac is "Here nightly observers scrawl in hasty abbreviations for the current weather conditions: clear weather [unicode Black Sun with Rays], cloudy [unicode cloud], rain [unicode umbrella], snow [unicode snowman], lightning [unicode lightning],

thunderstorm [unicode thunderstorm]."

The Danger Zone is below Alphabet Soup. The printed name of the Danger Zone is "[unicode skull and crossbones] Danger Zone [unicode skull and crossbones]".

The warning signs are in the Danger Zone. "A variety of international-standard warning standards suggest that this may not be the safest place: [unicode skull and crossbones], [unicode caution sign], [unicode radioactive sign], [unicode biohazard sign]."

This example text was used to produce a story file which has been tried against both Zoom for Mac OS X and Windows Frotz. The Latin, Greek, Cyrillic and Hebrew text all functioned perfectly on both, but a point of difference showed when writing the Hebrew alphabet: Zoom wrote this right-to-left, Windows Frotz left-to-right. The exotic symbols displayed on Zoom (though others not mentioned above, such as "[unicode staff of hermes]", did not): but most appeared only as black squares on Windows Frotz, exceptions being the astrological signs for Venus and Mars and the musical note.

7.1 **Example** **Fifty Ways to Leave Your Larva** *RB*

Using text substitution to make characters reply differently under the same circumstances.

We can use these substitutions to put together fairly complicated variations within a single piece of text:

"Fifty Ways to Leave Your Larva"

The Beekeeper's Palace is a room. Wasp is a woman in the palace. Drone is a man in the palace.

```
Instead of kissing someone:
    say "'[denial], [insult]! [boast]!'";
```

In this context, [denial] is understood to refer to the denial property of the noun -- but we could spell it out with "denial of the noun" if we wanted to.

```
A person has some text called denial. The denial of a
person is usually "Stand back". The denial of Drone is
"You forget yourself"

A person has some text called insult. The insult of a
person is usually "Grasshopper". The insult of Wasp is
"Larva".

A person has some text called boast. The boast of a
person is usually "I am ferocious". The boast of Drone is
"I have ferocious allies".
```

And then it would be trivial to insert further rules using these responses:

```
Instead of attacking someone:
    say "'Get away, [insult]!'"

Test me with "kiss wasp / hit wasp / hit drone / kiss
drone".
```

ExampleFifty Times Fifty Ways *RB*

Writing your own rules for how to carry out substitutions.

There is only so much we can cram into a text property, so being able to swap in properties is useful but limited. Fortunately, we can also, if we want, create new phrases for how to say things in brackets:

```
"Fifty Times Fifty Ways"


The Beekeeper's Palace is a room. Wasp is a woman in
the palace. Drone is a man in the palace.


A person can be fierce or mellow. Wasp is fierce. Drone
is mellow. A person can be calm or angry. A person is
usually calm. A person has some text called insult. The
insult of a person is usually "Grasshopper". The insult of
Wasp is "Larva".


Instead of kissing someone:
    say "'[denial for the noun], [insult for the noun]!
[boast]!'";
```

Now to provide some meaning to these bracketed forms. We'll start with the easy one:

```
To say boast:
    say "I have ferocious allies".
```

This is a "to say" phrase; we will learn more about phrases in a later chapter, but for now it may be enough to observe that whatever we write after "to say..." becomes a valid substitution in bracketed speech. In this particular case there is no advantage to using the boast token rather than spelling the text out in the quotation, but we might in theory add further instructions to randomize the output, for instance.

To say phrases can be more complex, as well, since we can have them incorporate extra information:

```
To say insult for (speaker - a person):
    if speaker is angry, say "[the insult of the noun]";
    otherwise say "small one".
```

Here where we have (speaker - a person), we are leaving a slot which

we can later fill in, madlibs-like, with any person we like. That is why we can write "insult for the noun": we are summoning the To say phrase and telling it to fill in the identity of the unknown speaker with the noun.

This differs from "insult of the noun" in the previous example; in that case, each person had his own insult property, and were merely printing that property out. Here we are actually telling Inform to calculate anew what the insult should be, and giving it some instructions about how to do that.

Our instructions can also get arbitrarily complex:

```
To say denial for (speaker - a person):
    if speaker is calm:
        say "You must not";
    otherwise if speaker is female:
        say "Stand back";
    otherwise:
        say "You forget yourself".


Instead of attacking someone:
    now the noun is angry;
    say "'Get away, [insult]!'"


Test me with "kiss wasp / hit wasp / kiss wasp / kiss
drone / hit drone / kiss drone".
```

So the effects we can get with text substitutions are quite flexible. We could even, if we wanted, fill in the substitutions by random choice, or by selecting items from a long list or table, should we have so bellicose a set of characters that they cannot make do with one or two insulting remarks apiece.

Chapter 6: Descriptions

6.7. Comparatives; *§6.8. Superlatives*; *§6.9. Which and who*; *§6.10. Existence and there*; *§6.11. A word*

Contents of *Writing with Inform*
Chapter 5: Text
Chapter 7: Basic Actions
Indexes of the examples

§6.1. What are descriptions?

It is in describing circumstances that Inform really capitalises on the concise, expressive power of natural language, and this chapter brings together the facts about "descriptions".

The simplest descriptions consist of a noun alone. Some refer to single things ("lantern", or "wine cask"), others to kinds of thing ("dead end" or "container"). But we have also seen adjectives alone:

```
The oaken desk is fixed in place.
```

Here, "fixed in place" is a description which, to Inform's simple-minded grammar, is a single adjective. And of course adjectives and nouns can be combined:

```
The cargo trunk is an openable container.
```

The description "openable container" consists of the noun "container", meaning a kind of thing, and the adjective "openable", which means one of the two possible states of an either/or property held by that thing.

As the next chapter will show, rules also make great use of descriptions:

```
Instead of throwing something at a closed openable door, say "Or
you could just use the handle like anyone else, of course."
```

We have already seen that we can list the items fitting a given description:

```
"You look down at [the list of things in the basket]."
```

It's also sometimes convenient to count them up:

number of (description of values) **... number**

This phrase counts the number of values matching the description, which may of course be 0. Example:

```
number of open doors
```

produces the number of doors, anywhere in the model world, which are currently open. A Problem message is produced if the number is potentially infinite, or impractical to count: for instance, Inform rejects "number of odd numbers".

It is because descriptions are so widely useful that they deserve a chapter of their own, and this is it.

Start of Chapter 6: Descriptions
Back to Chapter 5: Text: §5.13. Making new substitutions
Onward to §6.2. Adjectives and nouns

§6.2. Adjectives and nouns

Descriptions can contain a noun, but need not, and can contain any number of adjectives:

```
supporter = the noun supporter
closed = the adjective closed
the open wine cask = the adjective open + the noun wine cask
something portable = (some) + the noun thing + the adjective
portable
```

Note that we are not allowed to have more than one noun in the same description (something English occasionally does allow as a coded form of emphasis, as in "the man Jenkins" or "the harlot Helen").

Nouns are simple enough, referring either to kinds or specific things. The noun "something" means "some thing", so is actually a reference to the kind "thing". Inform treats this as having the same meaning as "anything", and all told there are eight special nouns of this kind, but with only three different meanings between them:

```
something = anything
someone = anyone = somebody = anybody
somewhere = anywhere
```

So for instance "anybody male" or "somewhere dark" are valid descriptions. These eight nouns are unusual in being allowed to come at the front of a description: nouns are usually expected to be at the end. (Inform also understands "nothing", "nowhere", "nobody", "no-one" and even "no one", which in a sense are opposites of "something" and the like, but for now we'll look at descriptions of things which do exist rather than don't.)

Start of Chapter 6: Descriptions

Back to §6.1. What are descriptions?

Onward to §6.3. Sources of adjectives

§6.3. Sources of adjectives

We have seen two sorts of adjectives so far: those which refer to either/or properties, like "open" and "closed", and those which come out of new kinds of value. If we define

```
Texture is a kind of value. The textures are rough, stubbly and
smooth. Everything has a texture.
```

...then "rough", "stubbly" and "smooth" all become adjectives. (That last sentence "Everything has a texture" was essential, because without it Inform would not know that these words could meaningfully be applied to things.)

In addition to these adjectives, we can create new ones (as we shall see), and a few special adjectives such as "visible", "touchable" and "adjacent" are already defined for

us by Inform.

Start of Chapter 6: Descriptions

Back to §6.2. Adjectives and nouns

Onward to §6.4. Defining new adjectives

§6.4. Defining new adjectives

Suppose we want to coin a word for supporters currently supporting something. We can do so with the following sentence:

```
Definition: A supporter is occupied if something is on it.
```

Note the colon, which is essential, and the usage of "it" in the definition part to refer to the object in question. (For this purpose we would write "it" even if we were defining a term about, say, a woman instead of a supporter, so that "she" or "her" might seem more appropriate - but see below.)

This creates the adjective "occupied", and gives it a definition valid for supporters. That restriction on validity means that non-supporters would always fail the description "something occupied"; which might be unfortunate if we wanted to talk about rooms being occupied. We could give a second definition thus:

```
Definition: A room is occupied if a person is in it.
```

These are entirely different senses of the word "occupied" - a mantelpiece is occupied if an invitation is on it, but for a drawing room to be occupied there must be human presence - and Inform applies whichever sense is relevant when deciding whether or not a given object is "occupied".

Often, though not always, we also want to give a name to the opposite possibility. We can do that as follows:

```
Definition: A room is occupied rather than unoccupied if a person is in
it.
```

The "rather than..." part of the definition is optional, but it saves having to write a

boringly similar definition of "unoccupied" out in longhand. (Note that Inform does not guess the meaning of "unoccupied" unless it has been explicitly told it. Such guesses are too risky, when so many "un-" words fail to conform to this pattern: "unified", "uncle", "ungulate" and so on.)

Newly defined adjectives cannot be used when creating things, because they are not explicit enough. Inform could not satisfy:

```
The Ballroom is occupied. The bucket is a large container.
```

because there is not enough information: by whom is the Ballroom occupied? How large, exactly? On the other hand, newly defined adjectives are very helpful in conditions and for rules, as we shall see later on.

It is occasionally clumsy having to refer to the subject of a definition using "it". We can avoid this and give the definition better legibility by supplying a name instead. For instance:

```
Definition: a direction (called thataway) is viable if the room thataway
from the location is a room.
```

which is a good deal easier to read than

```
Definition: a direction is viable if the room it from the location is a
room.
```

See **New conditions, new adjectives** *for giving more extensive definitions of new adjectives, using phrases*

Start of Chapter 6: Descriptions

Back to §6.3. Sources of adjectives

Onward to §6.5. Defining adjectives for values

Example 73: **Finishing School** The "another" adjective for rules such as "in the presence of another person".

§6.5. Defining adjectives for values

In general, any noun can have adjectives applied to it, and this means that values can have adjectives just as objects can. We have already seen that they can (in some cases, at least) have either/or properties, and this gives them adjectives just as for objects. But we can also write out definitions which apply to values:

```
Definition: A number is round if the remainder after dividing it by 10 is
0.
Definition: A time is late rather than early if it is at least 8 PM.
```

That makes the numbers 20 and 170 but not 37 meet the description "a round number", and the times 8 PM and 11:23 PM but not 9 AM meet the description "a late time". Because they come up fairly often, Inform contains several adjectives for numbers built in:

```
positive - one which is greater than zero (but not 0 itself)
negative - one which is less than zero (but not 0 itself)
even - a number like ..., -4, -2, 0, 2, 4, ...
odd - a number like ..., -5, -3, -1, 1, 3, 5, ...
```

Similarly, two useful adjectives are built in to talk about text:

```
empty - the text "", with no characters in it, not even spaces
non-empty - any text which does have at least one character in
```

Adjectives can have multiple definitions and, as long as each applies to a different sort of noun, there will be no problem. We could write:

```
A thing can be round, square or funny-shaped.
A container can be odd or ordinary.
```

And these definitions of "round" and "odd" will not interfere with the ones applying to numbers, because Inform can always look at the noun to see which definition is meant in any given case. For instance,

```
if the score is round, ...
```

must mean "round" in the sense of numbers, because the score is a number. Inform itself makes good use of this; "empty" also has meanings applying to rulebooks, lists and activities, for instance, as will be seen later.

Although it's more usual to give a definition to apply to a whole kind, we can actually give a specific definition to apply to just a single object or named value. For example:

```
A colour is a kind of value. The colours are red, green and blue.
Definition: red is subtle if the player is female.
Definition: a colour is subtle if it is blue.
```

The first definition of "subtle" takes precedence, of course, since it has the more specific domain - it applies only to red. The effect of this is that, if the player's female, the subtle colours are red and blue; if not, just blue.

Start of Chapter 6: Descriptions

Back to §6.4. Defining new adjectives

Onward to §6.6. Whereabouts on a scale?

Example 74: ★★★ **Only You...** Smoke which spreads through the rooms of the map, but only every other turn.

§6.6. Whereabouts on a scale?

Adjectives are often used in English to give a sense of where something is on a sliding scale. We talk about "a tall man" and "a short man", but without meaning that all men are either tall or short. If pushed, we might say that tall means about 6 feet and up, short means about 5 feet 6 and down, but we more often compare one person's height against another's.

Inform allows us to use adjectives in the same way. For example, every container has a number called its "carrying capacity". We can define:

```
Definition: A container is huge if its carrying capacity is 20 or more.
Definition: A container is large if its carrying capacity is 10 or more.
Definition: A container is standard if its carrying capacity is 7.
Definition: A container is small if its carrying capacity is 5 or less.
```

These definitions are similar to those in the previous section, but have a very specific (and strictly enforced) shape to them. The adjective must be a single word. We have to say "its" (i.e., of it), not the ungrammatical "it's"; we have to specify a property, and a literal value of it, and we must either give an exact value or else conclude with "or more" or "or less". If we create something with one of these properties:

```
The basket is a large container in the Shop. The thimble is a small
container in the Shop. The matchbox is a standard container in the
Shop.
```

then they will have the most moderate values they can have, that is, the basket will have carrying capacity 10 and the thimble 5 (and of course the matchbox 7). Both of the following tests will then fail:

```
if the basket is huge ...
if the basket is a small container ...
```

because the basket is neither huge nor small, but somewhere in between.

Sometimes the meaning of adjectives must depend on their context, as we see from the following example, where we assess heights in inches:

```
A person has a number called height. Definition: A man is tall if his
height is 72 or more. Definition: A woman is tall if her height is 68 or
more.
```

Inform then judges whether someone is or is not "tall" using different standards for men and for women, and

```
In the Shop are a tall man and a tall woman.
```

creates a man 72 inches tall and a woman 68 inches tall.

Start of Chapter 6: Descriptions

Back to §6.5. Defining adjectives for values

Onward to §6.7. Comparatives

§6.7. Comparatives

The special definitions in the previous section have a further effect. When we define:

> Definition: A container is large if its carrying capacity is 10 or more.

we not only say how to test if something is large (see if its capacity is at least 10) and how to create something large (give it a capacity of exactly 10), we also create a new form of comparison. Thus,

> if the basket is larger than the thimble ...
> if the thimble is not larger than the basket ...

are both true. If we also define "huge" and "small", as in the previous section, we also get comparisons "huger than" and "smaller than". Note that "huger than" has exactly the same meaning as "larger than": we can use whichever wording seems more natural. (For bacilli, for instance, we would probably not say "huger than", even though the meaning would be unambiguous.)

We can also compare two things to see if they share the same value of a property. For instance, to go back to the heights example, once we define "tall" and "short", we get that exactly one of the following will be true:

> if Adam is taller than Eve ...
> if Adam is the same height as Eve ...
> if Adam is shorter than Eve ...

Though it will not always seem natural wording, we can use the comparison "the same P as" for any property P which has a value. Do we think "if the basket is the same carrying capacity as the thimble" is good English? Maybe, maybe not. But we are always at liberty to spell things out in full:

> if the carrying capacity of the basket is the carrying capacity of the thimble ...

Start of Chapter 6: Descriptions

Back to §6.6. Whereabouts on a scale?

§6.8. Superlatives

Lastly, if we define an adjective in this calibrating way, we also automatically benefit from the use of the superlative form. That is, if we define

```
Definition: A container is large if its carrying capacity is 10 or more.
Definition: A container is small if its carrying capacity is 5 or less.
```

then we can talk about things like this:

```
the largest container
the smallest open container
```

Though we should be careful, in the second case, because we might get nothing: maybe all the containers are closed at the moment this is used. And in general there might be several equally large largest containers, in which case we should not rely on getting any particular one of those rather than another.

Note that Inform constructs comparatives and superlatives by a pretty simplistic system. If we want to use these forms for an adjective expressing the relatively large size of a room, we had better go with "roomy" (roomier, roomiest) - not "spacious" (spaciouser, spaciousest).

Start of Chapter 6: Descriptions
Back to §6.7. Comparatives
Onward to §6.9. Which and who

§6.9. Which and who

A description can not only talk about things in terms of themselves, but also in terms of their relationships to the rest of the world. For instance,

```
an open container on the table
a woman inside a lighted room
```

an animal carried by a man
a woman taller than Mark
something worn by somebody

are all valid descriptions. These are really abbreviations, having missed out the words "which is" or "who is", as appropriate:

an open container which is on the table
a woman who is inside a lighted room
an animal which is carried by a man
a woman who is taller than Mark
something which is worn by somebody

and indeed those are also valid descriptions. The other sentence verbs can all be used here, too. So for instance:

a man who does not wear anything
something which supports something

And sometimes we should spell out "who is" regardless:

a man who is not Sherlock Holmes

Since these clauses can be attached to the end of any valid description, descriptions can grow longer still:

something worn by a woman who is in a dark room

Pedants who flinch when "which" is used to introduce a restrictive clause are welcome to use "that" instead.

Start of Chapter 6: Descriptions
Back to §6.8. Superlatives
Onward to §6.10. Existence and there
Example 75: ★★ **Versailles** A mirror which will reflect some

§6.10. Existence and there

"There" is a curious word in English, which mostly refers to some place which is being talked about - but which can sometimes mean the whole world. In Ian Fleming's novel "From Russia With Love", a chapter narrating a committee meeting of SMERSH officers in Istanbul ends with one of the Russians saying:

```
There is a man called Bond.
```

What does this "there" mean? It really just means that Bond exists. In fact, he's watching the meeting through a concealed periscope, but the SMERSH general doesn't know that. All he is saying is that Bond is out there somewhere, and is not imaginary, or dead.

Inform also allows "there is" (or "there are") to talk about what exists, or does not. This is especially useful if, for some reason, we don't want to give a name to something. For example:

```
There is a door in the Summerhouse.
```

Another reason might be that we want to create something but not put it anywhere. If Inform reads the sentence:

```
There is a man called Bond.
```

then it creates a man, gives him the name Bond, but places him initially off-stage - not in any room, that is, but available to be brought into play later on, like an actor who is not needed until Act II.

"There" also provides a useful way to test what exists:

```
if there is a woman in the Summerhouse, ...
```

Or even:

```
if there is a woman, ...
```

which will be true if the model world contains even a single woman, on-stage or off. The alternative "there are" can also be used:

```
if there are women in the Summerhouse, ...
```

but note that this does not necessarily imply more than one woman is present, despite the plural. If we want that, we have to be more explicit:

```
if there is more than one woman in the Summerhouse, ...
```

or, of course, we needn't use "there is" at all:

```
if more than one woman is in the Summerhouse, ...
```

And we can also test non-existence:

```
if there is nobody in the Summerhouse, ...
if there is nothing on the mantelpiece, ...
```

Start of Chapter 6: Descriptions
Back to §6.9. Which and who
Onward to §6.11. A word about in

§6.11. A word about in

What does "in" mean? It's worth just a brief diversion to cover this, because "in" has two subtly different meanings.

Meaning 1. Usually, if X is "in" Y then this is because of containment. A croquet ball is "in" a croquet box, which is "in" the Summerhouse. This is the standard meaning, and is the one which happens if we write something like:

```
The croquet ball is in the box.
```

or if we ask a question like:

```
if the croquet box is in the Summerhouse, ...
```

This kind of "in" talks only about direct containment. If we ask

```
if the croquet ball is in the Summerhouse, ...
```

then the answer is that it isn't - it is in the box which is itself in the Summerhouse, but that's not the same thing.

This is almost always the meaning of "in" that we intend. This is only one of a number of relationships between objects - there are also "part of", "on", "worn by" and "carried by", for example. If we have

```
The bird feed is on the sundial.
```

...then "if the bird feed is in the sundial" won't be true: the relationship here is one called support (being on top of, in effect), not containment. But there's no confusion because "on" and "in" are different words, so it's no problem that they have different meanings.

Meaning 2. Much less common. If X is "in" Y and Y is a region, then the meaning is slightly different. Suppose the Garden Area is a region, and contains several rooms - the Croquet Lawn, the Terrace and so on. Then

```
if the croquet box is in the Garden Area, ...
if the bird feed is in the Garden Area, ...
if the Terrace is in the Garden Area, ...
```

are all true. This seems very natural, but in fact is quite different from the first meaning of "in". It allows rooms (and even other regions) to be "in" a region, and it allows indirect containment.

How Inform decides. So which meaning does Inform use, and when? Since these two meanings are so different, it clearly matters.

The answer is that meaning 1 is always the meaning of "X is in Y" unless Y is explicitly the name of a region. Thus:

```
if the croquet box is in the Garden Area, ...
```

is meaning 2, because "Garden Area" is the name of a region. That seems fair enough, but values are indeed sometimes given names (becoming "variables", or values "that vary"). Suppose "mystery value" is a name for a value which is an object, but which has different identities at different times. Then Inform reads

```
if the croquet box is in the mystery value, ...
```

as meaning 1, because whatever "mystery value" is, it isn't explicitly a region name, even if from time to time it might happen to be equal to a region.

That sometimes makes meaning 2 difficult to express. If we ever need it, and this is fairly rare, we can write it like so:

```
if the croquet box is regionally in the mystery value, ...
```

because "regionally in" is always meaning 2 of "in".

Start of Chapter 6: Descriptions

Back to §6.10. Existence and there

Onward to §6.12. A word about nothing

§6.12. A word about nothing

Like "in", "nothing" has two slightly different meanings, though here there's much less potential for confusion.

Meaning 1. "Nothing" as "no thing". This is the meaning in sentences like:

```
Definition: a container is bare if nothing is in it.
```

And similar for conditions like "if the box contains nothing". It's a word which describes the absence of things: it says that, though there might have been many

possible items here, it turned out that there were none.

Meaning 2. "Nothing" as a value. This is much less commonly seen, but sometimes Inform stores a value such as a property (or a variable) which always has to be an object. In some circumstances, "nothing" is then a special value meaning that this is not set at present. For instance,

```
Definition: a container is impossible if its matching key is nothing.
```

The "matching key" property of a container is always an object, but is allowed to be "nothing" when there isn't a matching key anywhere. (If such a container is locked, nobody will ever be able to unlock it.)

How Inform decides. So which meaning does Inform use, and when? The answer is that it depends on the relationship being talked about. When this is "is", values are being compared and we are using meaning 2. But when it is any other relationship, like "is in" - which talks about containment - then we are using meaning 1.

Start of Chapter 6: Descriptions
Back to §6.11. A word about in
Onward to §6.13. To be able to see and touch

§6.13. To be able to see and touch

Two of the adjectives built into Inform are:

```
"visible" - the player can see this
"touchable" - the player can touch this
```

So we can write descriptions such as "someone visible" or "a touchable container". We also have adjectives "invisible" and "untouchable", as might be expected. The visibility adjectives are particularly useful because the following is likely to go wrong:

```
if Helen is in a dark room, ...
```

This tests whether the room is dark, of itself; Helen may in fact be able to see by means of a torch, but the room is still "dark".

We can also talk about what other people can see and touch:

```
something which can be seen by Helen
```

are synonymous. Similarly for touch; and we can write such conditions as

```
if Helen cannot see Agamemnon, ...
if Cressida can see Troilus, ...
```

Note that it is essential to establish who does the seeing and touching: so "something which can be seen" will not be allowed, whereas "something which can be seen by Helen" will.

In fact, inside Inform the adjective "invisible" (for instance) has the following straightforward definition:

```
Definition: Something is invisible if the player cannot see it.
```

The exact definitions of visibility and touchability are complicated, because there are so many ways in which vision and touch can be obstructed, but the gist is that they behave as one would expect. Note that in darkness, nothing is visible, and that nobody can see from one room to another. In general anything invisible is also untouchable, but there are a few exceptions to do with being in the dark. Lastly, the player's own body (usually called "yourself" during play) is both visible (in light) and touchable.

Start of Chapter 6: Descriptions

Back to §6.12. A word about nothing

Onward to §6.14. Adjacent rooms and routes through the map

Example 76: ★★★ **Lean and Hungry** A thief who will identify and take any valuable thing lying around that he is able to touch.

---

§6.14. Adjacent rooms and routes through the map

Another useful adjective built into Inform is "adjacent". Two rooms are said to be adjacent if there is a map connection between them which does not pass through some barrier such as a door. This is easily tested:

```
if the Hallway is adjacent to the Study ...
```

We usually want to know about the places adjacent to the current scene of the action, so that is what the adjective "adjacent" means when applied to rooms. For instance:

```
if somebody is in an adjacent room, ...
```

As with the case of "visible", the adjective is a cut-down version of the more general relationship. This often happens: "worn" and "carried", for instance, imply "by the player" unless something else is specified.

If we want to ask a more direct question, we can obtain specific map connections as follows. (Recall that every map connection leads either to a door, to a room, or to nothing.) If we know which direction we want to look in, then the easiest thing is to use its relation - every direction in the map, say "north", has its own relation, say "mapped north of". So:

```
if the Ballroom is mapped north of the Hallway, ...
```

Alternatively, and particularly if the direction is not a constant,

room (direction) **from**/of (room) **... room**

This phrase produces the room which the given map direction leads to, or the special value "nothing" if it leads nowhere. If it leads to a door, the result is the room through that door. Examples:

```
say "You look north into [the room north from the Garden]."
if the room north from the Garden is nothing, say "The grass
leads nowhere."
```

door (direction) **from**/of (room) **... door**

This phrase produces the door which the given map direction leads to, or the special value "nothing" if it leads nowhere or to a room. Examples:

```
let the barrier be the door north from the Garden;
if the barrier is a door, say "Well, [the barrier] is in the way.";
```

room-or-door (direction) **from**/of (room) **... object**

This phrase produces the object which the given map direction leads to, which will always be either a room, a door or the special value "nothing". The phrase is used mainly by the Standard Rules, for technical reasons, and usually it's better to use "room ... from ..." or "door ... from ..." instead.

The map can be a great sprawling mass of rooms and doors connected together, and it can be quite hard to find a way through it one step at a time.

best route from (object) **to** (object) **... object**

This phrase produces a direction to take in order to get from A to B by the shortest number of movements between rooms, or produces "nothing" if there is no way through at all. Example:

```
The description of the brass compass is "The dial points
quiveringly to [best route from the location to the Lodestone
Room]."
```

Best routes are ordinarily forbidden to go through doors, but if the suffix "using doors" is added as an option then any open or openable and unlocked door may be used on the way; and if "using even locked doors" is given, then any door at all will do. Since magnetism is no respecter of property, that seems right here:

```
The description of the brass compass is "The dial points
quiveringly to [best route from the location to the Lodestone
Room, using even locked doors]."
```

In practice this simple approach sometimes produces impossible journeys, rather the

way Google Maps directions from New York to London would recommend driving down to the docks and then swimming. A more careful approach is to use:

best route from (object) **to** (object) **through** (description of objects) **... object**

This phrase produces a direction to take in order to get from A to B by the shortest number of movements between rooms which match the given description, or produces "nothing" if there is no way through at all. Example:

```
best route from the Drawbridge to the Keep through visited
rooms
```

The condition - in this case, that "visited rooms" must be used - also applies to both ends of the journey, so if either Drawbridge or Keep are unvisited then this is "nothing". (Similarly, saying something like "...through containers" would mean there is never a route.)

Lastly, the following phrases can find out how long the journey would be. (They are quite a bit faster than using the "best route..." phrases repeatedly and counting.)

number of moves from (object) **to** (object) **... number**

This phrase produces the number of map connections which must be followed in order to get from A to B by the shortest number of movements between rooms. If A and B are the same, the answer is 0; if there is no route at all, the answer is -1. Example:

```
The description of the proximity gadget is "You are now
[number of moves from the location to the Sundial] moves
from the Sundial.";
```

number of moves from (object) **to** (object) **through** (description of objects) **... number**

This phrase produces the number of map connections which must be followed in order to get from A to B by the shortest number of movements between rooms matching the given description. If A and B are the same, the

answer is 0; if there is no route at all, or if either A or B fail to match the description themselves, the answer is -1.

Route-finding makes it possible to write quite sophisticated conditions concisely. But these sometimes run slowly, because they call for large amounts of computation. How rapidly Inform can find routes depends on which of two methods it uses. Both have advantages - one is fast but needs large amounts of memory, the other is slow but economical. We can choose between them with one of these two use options:

```
Use fast route-finding.
Use slow route-finding.
```

If neither is specified, "fast" is used where the project uses the Glulx virtual machine (see the Settings panel), and "slow" on the Z-machine, where memory is tighter. Fast route-finding is ideally suited to situations where dozens of characters are constantly route-finding through the map as they meander around in a landscape.

See **Indirect relations** *for route-finding through a relation rather than the map*

Start of Chapter 6: Descriptions

Back to §6.13. To be able to see and touch

Onward to §6.15. All, each and every

Example 77: **Mistress of Animals** A person who moves randomly between rooms of the map.

Example 78: **All Roads Lead to Mars** Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

Example 79: **Hotel Stechelberg** Signposts such as those provided on hiking paths in the Swiss Alps, which show the

correct direction and hiking time to all other locations.

Example 80: ★★★ **A View of Green Hills** A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

Example 81: ★★★ **Unblinking** Finding a best route through light-filled rooms only, leaving aside any that might be dark.

---

§6.15. All, each and every

When testing conditions, we normally talk only about specific things, or else ask if a particular circumstance happens:

```
if the oaken door is open
if a woman is carrying an animal
```

But we can also use "all", "each" or "every" to check the whole range:

```
if each door is open
if anyone is carrying all of the animals
if everybody is in the Dining Room
```

Inform allows other English "determiners" (as they are sometimes called), as well:

```
if some of the doors are open
if most of the doors are open
if almost all of the doors are open
```

are true if at least one case is true, if a majority (any number greater than one half) or at least 80 per cent of the possible cases are true, respectively.

And we can also use "none" and "no". These three are all ways to say the same thing:

```
if no door is open
if all of the doors are not open
if none of the doors is open
```

though it may be clearer style to find a positive way of putting things:

```
if all of the doors are closed
```

All, each and every can be applied to values, too - but only in some cases. For example, suppose we write:

```
Colour is a kind of value. The colours are red, orange, yellow, green,
blue, indigo and violet. A colour can be found or unfound.
```

And suppose that, during play, we assign the "found" property to any colour which the player notices on a wall. We might then want to write conditions like so:

```
if every colour is found
if most of the colours are found
if any colour is found
```

But we always have to bear in mind that Inform might have no reasonable way to decide these questions. It will refuse to allow these, for example:

```
if every number is positive
if any text is palindromic
```

(even supposing the adjective "palindromic" has been defined) - there are practically infinitely many possible numbers and texts, so the search cannot sensibly be done.

Start of Chapter 6: Descriptions

Back to §6.14. Adjacent rooms and routes through the map

Onward to §6.16. Counting while comparing

Example 82: ★★★★ **Revenge**

§6.16. Counting while comparing

Lastly we can also ask for a more specific number of possibilities, like so:

```
if two women are carrying animals
if at most three doors are open
if fewer than 10 portable containers are closed
if all but two of the devices are switched on
if there are more than six locked doors
```

Likewise for "less than", "at least", "all except". Something to watch out for is that

```
if two doors are open
```

will be found true if there are (say) three open doors: after all, if three doors are open, then certainly two doors are. So this is not quite counting. We can be more precise by writing

```
if exactly two doors are open
```

The "all but" counts - say, "if all but two doors are open" - are exact: if, in fact, all of the doors are open then this will be found false.

We can often use these counting forms with values, too. As with the use of "all", this is allowed only if the kind of value is one which can reasonably be searched through. For example:

```
if more than three scenes are happening
if there are more than two non-recurring scenes
```

are allowed because the built-in kind of value "scene" (of which much more later on) has only a small number of possible values.

Lastly, note that the "the" in text like "two of the doors" matters: without it, the phrase

will not be recognised as a requirement on the number. (This is to make sure that names of things like "two of hearts" are not misinterpreted.)

Start of Chapter 6: Descriptions

Back to §6.15. All, each and every

Onward to Chapter 7: Basic Actions: §7.1. Actions

Example 83: ★★ **Yolk of Gold** Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

Examples from Chapter 6: Descriptions

Start of this chapter

Chapter 7: Basic Actions

Indexes of the examples

★ **Example** **Finishing School** *RB*

The "another" adjective for rules such as "in the presence of another person".

It is often useful to write action rules which apply only when the player is observed by a third party. "In the presence of a person", however, will react even if only the player is in the room, because the player is, of course, a person as well.

A convenient way around this problem is to define an "other" adjective:

"Finishing School"

The apple is an edible thing carried by the player.

```
The Hall is a room. Miss Wicket is a woman in the Hall.

The Dormitory is south of the Hall.

Definition: a person is another if it is not the player.

Instead of eating something in the presence of another
person:
    say "Your mannerly upbringing prevents you from
eating without a fork or knife in front of someone."

Test me with "eat apple / south / eat apple".
```

If we did not have "another" here, Inform would interpret even the player as a possible schoolmarm, leading to such lines as "yourself stares at you coldly...". Clearly not quite the thing.

ExampleOnly You... *RB*

Smoke which spreads through the rooms of the map, but only every other turn.

Suppose we want to have smoke that spreads from room to room, gradually filling the entire map with a clogging smoke. Having it spread every single turn would make for a pretty rapid diffusion, so we temper this by having it spread only on even-numbered turns, instead. Conveniently, Inform by default already knows about even and odd numbers, so we can write:

```
"Only You..."
```

Section 1 - The Procedure

Every turn when the turn count is even:
    if every room is smoky, make no decision;
    let previously smoky be whether or not the location is smoky;
    repeat with area running through smoky rooms:
        now every room which is adjacent to the area is smoky;
    if previously smoky is false and the location is smoky:
        say "[The location] is filling rapidly with smoke."

A room can be smoky or unsmoky.

Some air is a backdrop. Air is everywhere. Instead of doing something other than examining or smelling to air: say "It's just air." Understand "smoke" as the air when the location is smoky.

Instead of examining the air in a smoky room: say "A thick layer of smoke lies just under the ceiling."

Instead of smelling the air in a smoky room: say "Agh, acrid." Instead of smelling a smoky room: try smelling the air.

After looking in a smoky room: say "A thick layer of smoke has gathered under the ceiling."

Section 2 - The Scenario

The Guide Lodge is a room. "A very spacious room

capable of containing several hundred girls while they eat, talk, or do crafts. It is constructed in a not-unappealing rustic style, with floor-to-ceiling windows overlooking the lake below, and a fieldstone hearth at the center." The Guide Lodge is smoky.

The Kitchen is north of the Guide Lodge. "Multiple eight-burner ranges, ovens, and a walk-in refrigerator: you know the sort of thing."

The Industrial Pantry is east of the Kitchen. "Awe-inspiring quantities of food line every shelf, from the three-gallon tub of mayonnaise to the 50-pound tub of rice. Perhaps the most astonishing item is a bag of marshmallows big enough to double as a futon."

The player is in the Pantry.

The Hallway is west of the Guide Lodge. The description of the Hallway is "A perpetually-crammed hallway which has to handle the overflow line for the toilets." A singed sign is fixed in place in the Hallway. The description of the sign is "Where the edge of the sign has not been burnt, the legible words are '...Can Prevent Forest Fires'."

The Toilets are north of the Hallway. "Always in full use, at least when the 12-to-15s are here."

The Coat Closet is south of the Hallway. "Muddy boots may not be worn inside the lodge; instead, about 250 pair are piled here, along with their owners' damp parkas and umbrellas."

```
The Craft Supply Room is west of the Hallway. "A
holding-depot for jugs of white glue and popsicle sticks."


Test me with "x smoke / z / z / z / z / x smoke / look".
```

ExampleVersailles *RB*

A mirror which will reflect some random object in the room.

One of the advantages of descriptions is that we can use them to pick an item randomly from a specified category. (For more on this possibility, see the Change chapter sections on randomness.)

For instance, suppose we wanted to create a mirror in which the player would see some item from the room reflected. We might write

```
Instead of searching the mirror:
    say "You see [a random thing in the location]
reflected back at you."
```

This is the same as "a random thing which is in the location": phrase "in..." can be used briefly in Inform as it can in English.

But, on a little more thought, we might want to expand on this: the mirror perhaps should reflect not only things that are in the room, but anything that the player can see (even if it's on a supporter or carried by someone). So then we might instead write

```
Instead of searching the mirror:
    say "You see [a random visible thing] reflected back
at you."
```

There's still a risk, though, that this will produce the response

```
You see the mirror reflected back at you.
```

because, of course, the mirror is itself visible. So instead we might write

Instead of searching the mirror:
    say "You see [a random visible thing which is not the mirror] reflected back at you."

"Versailles"

The Hall of Plywood Boards is a room. "The Hall of Mirrors is under reconstruction: it is currently a dank tunnel enlivened only by short placards about the history of the room.

As though to mock tourists such as yourself who bought their tickets without knowing this, the officials have left uncovered a single panel of mirror."

The mirror is scenery in the Hall of Plywood Boards. Understand "panel" or "panel of mirror" as the mirror. The description is "Lovingly restored to shimmering brilliance, it suggests how marvelous this room would be if you had had the good sense to arrive after the renovations were complete."

Some tourists are a person in the Hall of Plywood Boards. The tourists are scenery.

Instead of searching the mirror:
    say "You see [a random visible thing which is not the mirror] reflected back at you."

Test me with "x mirror / look in mirror / g".

A final note: we use "searching" here because Inform understands both SEARCH THING and LOOK IN THING as the searching action, and the player is most likely to type LOOK IN MIRROR in order to see the reflection there. In the absence of an example, we can discover the relationship between actions and their command vocabulary in one of two ways. A complete list of actions and the vocabulary associated with them is available in the Actions index. Alternatively, we can type ACTIONS at a prompt, followed by LOOK IN MIRROR, and get the response

```
[searching the mirror]
You find nothing of interest.
[searching the mirror - failed the can't search unless
container or supporter rule]
```

...which tells us that Inform is understanding the action as "searching the mirror".

ExampleLean and Hungry *RB*

A thief who will identify and take any valuable thing lying around that he is able to touch.

```
"Lean and Hungry"

Substance is a kind of value. The substances are silver,
gold, and lead. Everything has a substance. A thing is
usually lead.

Definition: a thing is valuable if it is not lead.

The Limestone Cave is a room. "Not very big, and it
doesn't go back far, but you'd hoped to find some
shelter here. Outside it is raining exceptionally hard."
```

```
The sinister gentleman is a man in the Cave. "Leaning
against the wall is a sinister gentleman in a threadbare
waistcoat." The description is "He looks as though he
might once have been quite well off."


After examining the gentleman:
    say "[The noun] smiles back at you in an unnerving
fashion."
```

Now we make the rule that governs the gentleman's behavior. Here we're going to invoke the rules that allow characters besides the player to do actions. More about this can be found in the Advanced Actions chapter:

```
Every turn:
    if the sinister gentleman can touch something
valuable (called the treasure) which is not carried by a
person:
        try the gentleman taking the treasure.


Report the gentleman taking something:
        say "[The gentleman] slyly acquires [the noun] and
tucks it into his pocket." instead.
```

That "not carried by a person" prevents the gentleman from stealing from the player (or, less plausibly, from himself). If we did want him to pick pockets, we could just have said "which is not carried by the gentleman".

```
The player is carrying a coin, a bust of Abraham Lincoln,
a bottle of sherry, a small pistol, and a wad of
Confederate cash. The coin is gold. The pistol is silver.


A locket is in the Cave. The locket is gold. "A broken
gold locket lies on the ground, a token of your late
Mama."
```

```
Test me with "get locket / i / drop locket / z / drop cash /
z / x gentleman / drop all / z / z / look".
```

---

77 Example Mistress of Animals *RB*

A person who moves randomly between rooms of the map.

Suppose we want a restless sort of character always pacing from room to room. It is quite easy to use adjacency to achieve this effect:

```
"Mistress of Animals"


Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.


Artemis is a woman in Thebes.


Every turn:
    if Artemis is in a room (called the current space):
        let next space be a random room which is adjacent
to the current space;
        if Artemis is visible, say "Artemis heads to [the next
space].";
        move Artemis to next space;
        if Artemis is visible, say "Artemis arrives from [the
current space]."


Test me with "z / z / z / z / z / z".
```

Of course, it helps that Artemis is the sort to like open spaces. The

implementation would become more complicated if there were doors which might block transit between these locations.

Example **All Roads Lead to Mars** *RB*

Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

Suppose we want to allow the player to wander freely in any direction, but ourselves maintain control over the order in which he encounters the rooms. This sort of effect emphasizes the order of the story-telling over any kind of rigorous simulation of space; on multiple play-throughs, the player might not find all the same rooms in the same locations.

```
"All Roads Lead to Mars"


Before going a direction (called way) when a room
(called next location) is not visited:
    let further place be the room the way from the
location;
    if further place is a room, continue the action;
    change the way exit of the location to the next
location;
    let reverse be the opposite of the way;
    change the reverse exit of the next location to the
location.


The Open Plain is a room. "A wide-open grassy
expanse, from which you could really go any way at all."


The Hilly Place is a room. "The grassland gives way to a
somewhat more hilly area, though there is still very little
to guide you any particular way."


The Stream is a room. "This is the third place you've
```

```
been today, and so the stream is welcome. How
refreshing!"



Test me with "n / s / e / e".
```

If we wanted still to be able to find routes between places, we could define a relationship of connection between rooms, which we would add to as we went along.

ExampleHotel Stechelberg *RB*

Signposts such as those provided on hiking paths in the Swiss Alps, which show the correct direction and hiking time to all other locations.

The following rule appends a paragraph to every room description. We need not worry about doors (despite the pass in the Bernese Oberland known figuratively as the "Little Door").

```
"Hotel Stechelberg"



After looking:
    say "Yellow arms on the signpost point:-[line break]";
    repeat with destination running through interesting
rooms:
        let the way be the best route from the location to
the destination;
        if the way is a direction, say " [way] for [the
destination]: [number of moves from the location to the
destination] Std."



Hotel Stechelberg is a room. "The wooden hiking inn at
the end of the road, with flowerboxes, canton flags,
outdoor tables and a triangular paddock for the cows
contesting the annual Miss Stechelberg competition.
Otto and Marianne do cheerful innkeeper things, while
the sun blazes from a gentian-blue sky."
```

A room can be dull or interesting. A room is usually dull.

North of Hotel Stechelberg is Trummelbachfalle. North of Trummelbachfalle is Lauterbrunnen. Lauterbrunnen is interesting.

Southeast of Hotel Stechelberg is Trachsellauenen. Trachsellauenen is interesting.

Test me with "look".

With a bit more work, the result might be:

Hotel Stechelberg
The wooden hiking inn at the end of the road, with flowerboxes, canton flags, outdoor tables and a triangular paddock for the cows contesting the annual Miss Stechelberg competition. Otto and Marianne do cheerful innkeeper things, while the sun blazes from a gentian-blue sky.

Yellow arms on the signpost point:-
north for Lauterbrunnen: 2 Std.
west for Sefinental: 2 Std.
west for Schilthorn: 6 Std.
southeast for Trachsellauenen: 1 Std.
southeast for Oberhornsee: 3 Std.

ExampleA View of Green Hills *RB*

A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

Suppose a game in which the player is wandering an open landscape with long vistas, allowing him to LOOK in some direction, or even look at an adjacent location.

```
"A View of Green Hills"


Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.


Understand "look [direction]" as facing.


Facing is an action applying to one visible thing.


Carry out facing:
    let the viewed item be the room noun from the
location;
    if the viewed item is not a room, say "You can't see
anything promising that way." instead;
    try looking toward the viewed item.
```

In rules about action handling, "noun" refers to the first object that the player has mentioned in his command, so if the player typed >LOOK WEST, "let the viewed item be the room noun from the location" would be processed as "let the viewed item be the room west from the location", and so on.

We can at need override the default behavior, if it is not going to be appropriate for the player to see the next room over. There is only sky above at any time, so...

```
Instead of facing up:
    say "Above you is bright sky."
```

```
Understand "look toward [any adjacent room]" as
looking toward. Understand "examine [any adjacent
room]" as looking toward.


Looking toward is an action applying to one visible thing.


Carry out looking toward:
    say "You make out [the noun] that way."
```

This design allows us to create descriptions for rooms (as seen from the outside) which will work regardless of where we're looking from. For instance:

```
Instead of looking toward Athens:
    say "Even from here you can make out the silhouette
of the Acropolis."


Test me with "look north / look south / look up / look
east / east / look west".
```

ExampleUnblinking

Finding a best route through light-filled rooms only, leaving aside any that might be dark. *RB*

Suppose we're simulating a situation where the player needs to travel through lit areas only, but we want to give him some hints about which way might be safe. Here we'll find our best route through light-filled rooms.

The slightly tricky part is that it's not necessarily easy to tell whether a room has a lamp in it. We may say "if the Crypt is lighted", but that only tells us whether it has been declared to be inherently lighted or dark, not whether it happens to contain a light source that the player would be able to see if he went in.

The easiest way to get around this is to create an object -- the light-meter; place it in the target location; and check whether it "can see" a lit object. This preserves all the usual rules about open and closed containers, transparency, etc.

"Unblinking"

Section 1 - Procedure

The light-meter is a privately-named scenery thing.

```
Definition: a room (called the target room) is light-filled:
    if the target room is lighted:
        yes;
    move the light-meter to the target room;
    let the answer be false;
    if the light-meter can see a lit thing:
        now the answer is true;
    now the light-meter is nowhere;
    decide on the answer.
```

That done, we're free to use our best-route phrases to choose a particular route.

Section 2 - Scenario

The Tomb of Angels is a room. "This ancient underground tomb is lightless but for a few shafts from the surface. Everywhere in the shadows are carved angels, their faces worn away by water and pollution, their wings little more than nubs."

The Upward Path is above the Tomb of Angels. It is dark. "The staircase switches back on itself many times

```
as it ascends towards the Crash Site."

A container called the sarcophagus is in the Upward
Path. It is closed and openable. "A sarcophagus rests in
the niche here, [if open]the lid pushed
aside[otherwise]the lid firmly in place[end if]."

The Crash Site is above the Upward Path. "The ceiling
has wholly caved in here, and the belly of the spaceship
above you is visible -- including the escape hatch."

A candle is a kind of thing. A candle is usually lit. The
player carries four candles.

After looking when the location is not the Crash Site:
    if the best route from the location to the Crash Site
through light-filled rooms is a direction (called next way):
        say "It looks like there's a safe, lit path [if the
number of moves from the location to the Crash Site
through light-filled rooms is 1]straight[otherwise]if you
go[end if] [next way].";
    otherwise:
        say "It looks like there is no fully lit path from here
to the Crash Site."

Test me with "up / drop candle / down / up / take the
candle / open sarcophagus / put candle in sarcophagus
/ down / up / close sarcophagus / d".
```

An important word of caution: this method would give false negatives if there were a backdrop lightsource, such as the moon, providing light to the Upward Path. This is because backdrops are actually moved around the map by Inform during play, following the player around. So if the moon backdrop is in the Crash Site with the player, it will not be in the Upward Path as well -- even if it's scheduled to move there as soon as the player does.

This example does include a number of features that we haven't met yet, particularly rulebooks. Nonetheless, the basic idea should be relatively clear. Our premise is that the player is faced with a series of disgruntled furnishings, none of which want to be responsible for supporting the cold, hot, drippy, or spiky objects lying around the room.

"Revenge of the Fussy Table"

Use scoring.

Section 1 - The Supporters

The Dining Room is a room. "A room of handsome proportions, with an octagonal dining table on a fine parquet floor. The original set of chairs has been taken away, leaving only a red velvet dining chair and a blue suede armchair."

The fussy table is a supporter in the Dining Room. It is scenery. The description is "It's hypochondriac: the result of a sojourn in a superior antique shop, where it picked up the scent of almond oil, words like provenance, and a secretive shame about being resurfaced.

Chairs don't get along with it.

In its mellowest moods, after a really good dusting, it will

mention, casually, a cabinet it once knew, which was twelve feet tall, came from the Winter Palace, and had an asking price of $350K."

A chair is a kind of supporter with carrying capacity 1. A chair is enterable. A chair is usually scenery. Understand "chair" as a chair.

The blue suede armchair and the red velvet dining chair are chairs in the Dining Room. The description of the blue suede is "Quiet, mellow, and with a lingering aroma of clove cigarettes, and possibly something less legal. It doesn't sit up straight, and doesn't let anyone else do so either." The red velvet has the description "In temperament, in bearing, in everything it is the opposite of the blue suede chair. It is concerned for its duty all the time, has a rigidly straight back, and pokes diners in the spine when it suspects them of overdoing things."

The carrying capacity of the player is 2.

Section 2 - Sources of Unpleasantness

A thing can be drippy or dry.

Heat is a kind of value. The heats are luke-warm, cold, and scalding. Everything has a heat. Understand the heat property as describing a thing.

A thing can be spiky or smooth. The cleated left shoe is a wearable spiky thing. It is worn by the player. The cleated right shoe is a wearable spiky thing. It is worn.

Definition: a thing is unpleasant if it is scalding or it is cold or it is spiky or it is drippy.

Definition: a supporter is occupied if something is on it.

Definition: something is contented if it is not concerned.

Definition: a chair is concerned if something unpleasant is on it.

Definition: the table is concerned if something drippy is on it or something scalding is on it or something spiky is on it.

Definition: the ceramic tile is concerned if something spiky is on it.

Definition: a room is concerned if something drippy is in it or something scalding is in it or something spiky is in it or someone which is in it is wearing a spiky thing.

Before printing the name of a drippy thing: say "drippy ".
Before printing the name of a cold thing: say "cold ".
Before printing the name of a scalding thing: say "scalding ".

Section 3 - What the Player Can Do About It

After putting a cold thing on a scalding thing:

say "[The noun] meets [the second noun]; both shriek in pain. But the necessary heat transfer occurs.";
now the heat of the noun is luke-warm;
now the heat of the second noun is luke-warm.

After putting a scalding thing on a cold thing:
say "[The noun] meets [the second noun]; both shriek in pain. But the necessary heat transfer occurs.";
now the heat of the noun is luke-warm;
now the heat of the second noun is luke-warm.

After taking a scalding thing:
say "'Right,' you say. 'I'll just hold onto [the noun], shall I? Because that won't be in the least uncomfortable for me.'"

After taking a drippy thing:
say "With a sense of martyrdom, you retrieve [the noun], which dribbles water down your cuffs."

After entering a chair when the player is wearing a spiky thing (called the clog):
say "You sit on [the noun], lifting [the list of spiky things worn by the player] so that the parquet is no longer affronted."

Section 4 - In Which the Furnishings Complain

An every turn rule:
follow the complaint rules.

The complaint rules is a rulebook.

```
A complaint rule:
    if something (called the offending item) on the table is
drippy:
        say "'Help! Get me a coaster!' screams the table[if
the table is visible], its veneer squirming under [the
offending item][otherwise] from the Dining Room[end
if].";
        rule succeeds;
    if something (called the offending item) on the red
chair is drippy:
        say "'Oh dear,' murmurs the red chair, as [the
offending item] drips into its velvety seat. 'Oh dear, I will
have a damp spot. This is so very -- what will people
think?'";
        rule succeeds;
    if something (called the offending item) on the visible
armchair is drippy:
        say "[The offending item] visibly begins degrading
the suede where it sits. The armchair is tactfully silent.";
        rule succeeds;
    if a drippy thing (called the offending item) is in the
location and the player is in the Dining Room:
        say "'Cripes,' says the parquet. 'No one mind me at
all. Just leave that [offending item] right here. You know
I'm the most valuable thing in the room?'";
        rule succeeds.



A complaint rule:
    if a cold dry thing (called the offending item) is on the
table:
        say "The table shivers under [the offending item].";
        rule succeeds.



A complaint rule:
    if something (called the offending item) on the table is
scalding:
        say "'Hey!' protests the table, practically smoking at
[the offending item]. 'Heard of a trivet?'";
        rule succeeds;
    if something (called the offending item) on the
armchair is scalding:
        say "The armchair doesn't complain about [the
offending item]; it just begins, quietly, to give off warm
sweet clouds of blue steam, as though its inner opium-
nature had been released.";
        rule succeeds;
```

```
    if something (called the offending item) in the Dining
Room is scalding:
        say "'Yes indeed,' [if the location is the Dining
Room]says[otherwise]bellows[end if] the parquet. 'That
[offending item] won't leave any sort of mark what-so-
ever.'";
        rule succeeds.



A complaint rule:
    if something (called the offending item) on the table is
spiky:
        say "The table holds very very very still lest [the
offending item] mar its finish. But its resentment is
palpable.";
        rule succeeds;
    if something (called the offending item) on the velvet
chair is spiky:
        say "[The offending item] stabs the velvet of the
red velvet chair, which draws itself up even more
sharply in a pose of dutiful martyrdom.";
        rule succeeds;
    if something (called the offending item) which is spiky
is on the ceramic tile:
        say "'[The offending item] itches!' cries the ceramic
tile.";
        rule succeeds;
    if something (called the offending item) in the location
is spiky:
        say "'[The offending item] should not be on a
parquet floor,' sings the parquet floor sadly.";
        rule succeeds;
    if someone (called the offending person) in the
location is wearing something spiky and the location is
the Dining Room:
        say "The parquet floor mutters about the things
people wear to walk on parquet floors these days.";
        rule succeeds.



A complaint rule:
    if the table supports something which supports a
drippy thing (called the offending item),
        say "'[The offending item] could still theoretically
dribble down here,' says the table nonchalantly. 'It's
possible.'";
    if the velvet chair supports a concerned thing (called
the friend),
```

```
        say "'[The friend] seems a little worried,' remarks
    the velvet chair helpfully."
```

```
A complaint rule:
    if the concerned table can see a supporter (called the
    object of envy) which is not concerned:
        say "The table looks jealously at [the object of
    envy]. 'Some people have such an easy life.'";
        rule succeeds.
```

```
Instead of going to an adjacent room when the player is
    in the Dining Room:
    say "'You're not leaving!?' whimpers the table at
    once."
```

Moreover, every time the player gets close to resolving this issue, his unhelpful companion Alison brings in something else inconvenient. We can use the counting of contented supporters to decide when the player is getting close to winning and it's time for her to bring something else...

```
Section 5 - Source of Further Complications
```

```
The Tidy Kitchen is south of the Dining Room. Alison is
    a woman in the Tidy Kitchen. The kettle is a scalding
    thing. The ceramic tile is a cold portable supporter. It
    has carrying capacity 1. The platter is a scalding
    portable supporter. It has carrying capacity 1. The glass
    of ice water is a cold, drippy thing.
```

```
Instead of drinking the glass of ice water, say "It's not
    yours, it's for Alison, once she finishes cooking."
```

```
Alison carries the kettle, the tile, the platter, and the
    glass.
```

```
An every turn rule:
    follow the behavior rules.


The behavior rules is a rulebook.


A behavior rule:
    if ice water is carried by Alison and the player is in the
Dining Room and almost all of the supporters are not
concerned:
        let the next victim be a random fixed in place
contented supporter;
        move the ice water to the next victim;
        say "Humming to herself, Alison brings in [an ice
water] and sets it down smack on [the next victim], then
goes back out.";
        rule succeeds.


A behavior rule:
    if platter is carried by Alison and the player is in the
Dining Room and almost all of the supporters are not
concerned:
        let the next victim be the red velvet chair;
        move the platter to the next victim;
        say "Whistling a jaunty tune, Alison brings in a hot
platter and deposits it on [the next victim], then returns
to the kitchen.";
        rule succeeds.


A behavior rule:
    if the tile is carried by Alison and the player is in the
Dining Room and almost all of the supporters are not
concerned:
        let the next victim be a random contented chair;
        move the tile to the next victim;
        say "Humming to herself, Alison brings in [the tile]
and sets it down smack on [the next victim], then goes
back out.";
        rule succeeds.
```

```
A behavior rule:
    if kettle is carried by Alison and the player is in the
Dining Room and at least three fixed in place supporters
are contented:
        let the next victim be the fussy table;
        move the kettle to the next victim;
        say "Just then the kettle boils in the kitchen.
Whistling chirpily, Alison brings it in and sets it down
smack on [the next victim], then goes back out.";
        rule succeeds.
```

And just so that the player knows where he stands at the end of each turn:

```
Section 6 - General Assessment
```

```
An every turn rule:
    if some of the things are concerned, say "You sense
some resentment from [the list of concerned things]."
```

```
An every turn rule:
    if the player is carrying more than one scalding thing:
        let the dropped item be a random scalding thing
carried by the player;
        say "You can't hang onto so many burning hot
things, and lose your grip on [the dropped item].[line
break]";
        silently try dropping the dropped item.
```

```
An every turn rule:
    if almost all of the supporters are concerned,
        say "'I hope everything is going well,' bellows
Alison from the other room."
```

Finally, we need some rules to decide when the player is, in fact, finished.

```
Section 7 - Scoring
```

An every turn rule:
    now score is 5 minus the number of concerned things;
    if the location is concerned, decrement the score;
    if all of the supporters are concerned and the location is concerned, end the story;
    if none of the supporters are concerned and the location is not concerned, end the story finally.

The maximum score is 5.

When play begins, now score is 4.

Test me with "i / take off left shoe / get the glass / put the glass on the platter / get the tile / drop the tile / get the kettle / put the kettle on tile / sit on the blue armchair".

Example Yolk of Gold *RB*

Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

Suppose that for dramatic effect we would like the player to find the thing he is looking for always in the last drawer he opens...

"Yolk of Gold"

The Turret is a room. "A cramped little room distinguished chiefly by the spiral staircase that descends from it. The windows look out over the rooftop."

The Rooftop is outside from the Turret.

The staircase is an open unopenable door. It is scenery. It is below the Turret and above the Library. The description is "A winding stair carved out of the single trunk of a massive tree, all in a dark wood; the outside of the stairs left unfinished with the bark still on, but the treads worn smooth by long and constant use."

The description of the Library is "Here, it seems, you have found your mark: books line both walls, a dark carpet lies on the floor, and a strange dress hangs up in a presentation case. And the thing you were told to look for, a desk with three drawers.

A spiral staircase leads up and out."

The cherry desk is scenery in the library. The description is "A deep, satin-lustrous cherry, with scrollwork legs and gilt touches. The years have not been kind, and it has cracked and split in several places; the finish is damaged, and where there is inlaid mother of pearl, it is beginning to come up from its bed. But it is still a sound piece, and features three drawers."

After examining the desk for the first time, say "(Your employers were able to tell you to look for it, but not which drawer to look in. Typical.)"

A drawer is a kind of container. A drawer is always openable and closed. The description of a drawer is "The usual drawer of heavy wood, inadequately smoothed for ease of use."

The top drawer is a drawer. The middle drawer is a drawer. The bottom drawer is a drawer. The top drawer, the middle drawer, and the bottom drawer are part of the desk. A drawer can be explored or unexplored. A drawer is usually unexplored. Instead of searching a closed drawer, try opening the noun.

After opening a drawer when no drawers are explored:
    now the noun is explored;
    say "There is a tremendous screech, but nothing whatsoever inside, not even dust."

After opening an unexplored drawer when exactly one drawer is explored:
    now the noun is explored;
    say "You struggle to open this one a bit more quietly, conscious all the time of noise... but no, it's empty. Just stands to reason."

After closing a drawer:
    say "There is a racket of wood protesting against wood as you do so, which makes you wonder if you hadn't better just leave them open from now on. The unhappy owner is probably going to catch on soon enough anyway."

Before opening an unexplored drawer when exactly two drawers are explored:
    move the mysterious thing to the noun;
    now the noun is explored.

There is a mysterious thing. The description is "A very familiar-looking hemispherical weight of metal, goldish in tone though perhaps not made of gold. This one has a slight stickiness about the bottom surface."

```
Instead of opening the desk when at least one drawer is
unexplored:
    let the next drawer be a random unexplored drawer;
    say "(starting with [the next drawer])[line break]";
    try opening the next drawer.


Instead of looking under the desk when no drawers are
explored:
    say "You carefully survey the ground around the
desk. There don't seem to be any hidden tripwires or
traps to prevent you from having a look in the drawers."


Instead of looking under the desk when at least one
drawer is unexplored:
    say "Nothing there either. Thoroughness is a virtue
with tedium as a side effect, as your mother used to say
-- but they always counted her a trifle slapdash."


Instead of looking under the desk when all drawers are
explored:
    say "There's nothing down there, but this doesn't
come as a great surprise."


Instead of searching the desk when at least one drawer
is unexplored:
    move the mysterious thing to the player;
    say "You perform a hasty, squeaky search of ";
    if no drawer is explored, say "all three drawers,
discovering nothing and nothing in rapid succession. But
on the third drawer you ";
    if exactly two drawers are unexplored, say "the
remaining two drawers. There's nothing in the second,
but in the third you ";
    if exactly one drawer is unexplored, say "the last
drawer. In it, you ";
    say "turn up a promising hemispherical object.";
    now all the drawers are explored.
```

Perhaps, just for fun, we have all the other scenery draw the player's

attention back to the main point, as well.

The carpet is scenery in the Library. The description is "It is too dark for you to make out details of the antique design, which seems dimly to represent an early voyage to the moon, with ships and the travelling stars." Understand "rug" as carpet.

Instead of looking under the carpet:
    if none of the drawers are explored, say "You peel up a corner of the rug gingerly; nothing results. A perfectly ordinary rug, then." instead;
    if all of the drawers are explored, say "No time for that kind of nonsense. You'd better get out and away while you can." instead;
    if some of the drawers are explored, say "No time for that nonsense. The desk's what you want now; what you came for won't be embedded in the flooring." instead.

The glass presentation case is transparent scenery in the Library. The description is "The case is taller than you are, framed in wood, with large panels of glass, the better to display the remarkable contents."

Instead of attacking the glass presentation case:
    say "The glass pane of the presentation case shatters, throwing fine glass everywhere, including over the delicate museum-piece inside. Nor does the noise pass unnoted: only a second passes before there are footsteps in the hall, and though you make for the concealing darkness and escape of the turret, you are not swift enough. The servants are soon on you, and you are made to regret, quite painfully, this casual act of vandalism.";
    end the story saying "You have lost your opportunity."

The strange dress is a wearable thing in the glass case. The description is "Not the sort of dress that anyone

wears now: such elaboration would be ludicrous. It drips gold -- gilt lace, ruffles of trimmed gold, shimmering golden tracery -- dulled here and there by the sinister black of faceted jet."

Test me with "d / x case / x dress / x carpet / look under carpet / look under desk / x desk / open top drawer / close top drawer / look under desk / open bottom drawer / close bottom drawer / open middle drawer / get thing / look under carpet / look under desk / up".

Chapter 7: Basic Actions

and exceptional actions; *§7.10. The noun and the second noun*; *§7.11. In rooms and regions*; *§7.12. In*

Contents of *Writing with Inform*
Chapter 6: Descriptions
Chapter 8: Change
Indexes of the examples

§7.1. Actions

"Actions" are what we get if we try to break down a narrative into its irreducible parts. We might casually say that we are "going shopping", but this involves many smaller steps: going north, going east, entering the shop, examining a loaf of bread, taking it, giving money to the baker, and so on.

An action is an impulse to do something. This may or may not be a reasonable aspiration, and may or may not be achieved. The player's exploration of an interactive fiction is made by a sequence of actions, so much of the designing process comes down to responding to these actions.

We write actions using present participles. For instance, if the player types "take napkin" or "get the napkin" or something similar then the resulting action would be written as:

```
taking the napkin
```

The details of what words the player actually typed are unimportant to us: we deal only in actions.

Every action ends in success or failure. In this context, success means only that the player's intention has been fulfilled. If the player sets out to take the napkin, but finds a million-pound banknote in its folds instead, the action will be deemed to be a failure.

The testing command ACTIONS causes Inform to log every action as it happens, and what its outcome is. (ACTIONS OFF turns this off again.) For instance:

```
>s
```

```
[going south]
Security Vault
You can see a metal door here.
[going south - succeeded]
>close door
[closing metal door]
You close the metal door.
[closing metal door - succeeded]
>take door
[taking metal door]
That's fixed in place.
[taking metal door - failed the can't take what's fixed in place rule]
```

A good way to get a sense of the constant flow of actions is to use this command and then wander around an existing work, trying things out. ACTIONS can also give an insight into the web of rules governing play: there are more than ten different ways an attempt to take something can fail, for instance.

Start of Chapter 7: Basic Actions

Back to Chapter 6: Descriptions: §6.16. Counting while comparing

Onward to §7.2. Instead rules

§7.2. Instead rules

An action is ordinarily handled by running it through Inform's extensive rulebooks of what might be called normal behaviour. An action such as "taking the napkin", for instance, will be run through numerous checks to see if it is physically reasonable, and then provided all is well, the napkin will be moved into the possession of the player.

Instead, though, we can bypass the rules to do with an action and do something else:

```
Instead of eating the napkin: say "Why not wait for the actual dinner
to arrive?"
```

This is an example of a "rule": a set of circumstances followed by a list of instructions. When those circumstances apply, the instructions are carried out. In the case of an "instead" rule, after this is done the action is immediately ended (and counts as a failure, since the original intention has been thwarted).

A friendly alternative can be used when there is only a single instruction, as here: in such rules the colon can be replaced with a comma. Thus:

```
Instead of eating the napkin, say "Why not wait for the actual dinner
to arrive?"
```

Start of Chapter 7: Basic Actions

Back to §7.1. Actions

Onward to §7.3. Before rules

Example 84: ★ **Grilling** A grill, from which the player is not allowed to take anything lest he burn himself.

Example 85: ★ **Bad Hair Day** Change the player's appearance in response to EXAMINE ME.

§7.3. Before rules

Despite what was said in the previous section, instead rules do not quite bypass all of the usual rules. Inform knows that certain actions require light: for instance,

```
examining the napkin; looking; looking under the dining table
```

and if it is dark then none of these actions will be allowed, and any instead rules about them will not even be reached. Similarly, Inform knows that most actions require physical access to their objects: so "taking the napkin" would be blocked if the napkin were, say, inside a closed glass bottle, whereas "examining the napkin" would not. So an instead rule can only take effect if the action has already passed these basic reasonability tests.

"Before" rules genuinely precede checking of any kind. They also differ from instead rules in that they do not automatically stop the action in its tracks. Rather, they are provided as an opportunity to ensure that something else is done first. For example:

```
Before taking the napkin, say "(first unfolding its delicate origami
swan)".
```

whence

```
>GET NAPKIN
(first unfolding its delicate origami swan)
Taken.
```

We have seen that instead rules automatically stop actions, whereas before rules automatically allow them to continue. We sometimes want to change this. The magic word "instead" can therefore be tacked on to any instruction in a before rule, and will have the effect of immediately stopping the action at that instruction. Thus the following two rules are (almost) equivalent:

```
Before taking the key, instead say "It seems to be soldered to the
keyhole."
```

```
Instead of taking the key, say "It seems to be soldered to the
keyhole."
```

It is also possible to be explicit about stopping the action:

stop the action

This phrase stops the current rule, stops the rulebook being worked through, and finally stops the action being processed. Example:

```
Before taking the key:
    say "It seems to be soldered to the keyhole.";
    stop the action.
```

Finally, we can prevent Inform from stopping the action when it otherwise might:

continue the action

This phrase ends the current rule, but in a way which keeps its rulebook going, so that the action being processed will carry on rather than being stopped. Example:

```
Instead of taking the napkin:
    say "(first unfolding its delicate origami swan)[command
clarification break]";
    continue the action.
```

An "instead" rule ordinarily stops the action when it finishes, so the "continue the action" is needed to make things carry on. (This rule would have been better written as a "before" rule, in fact, but it shows the idea.)

As a general principle, it is good style to use instead rules whenever blocking actions, and before rules only when it is genuinely necessary to do something first but then to continue: in fact, it is good style to use "stop the action" or "continue the action" as little as possible.

Start of Chapter 7: Basic Actions

Back to §7.2. Instead rules

Onward to §7.4. Try and try silently

Example 86: ★★★ **Democratic Process** Make PUT and INSERT commands automatically take objects if the player is not holding them.

Example 87: ★★★★ **Sand** Extend PUT and INSERT handling to cases where multiple objects are intended at once.

§7.4. Try and try silently

Chapter 2 noted that surveys of Inform source text showed that the three most popular phrases used by authors are "say", "if" and "now". The fourth most popular is "try", which allows us to trigger off actions ourselves, rather than waiting for the player to type something which generates them. Thus:

try (action)

This phrase makes the action, which has to be named literally, take effect now. Example:

```
Instead of entering the trapdoor, try going up.
```

It's as if the player had typed GO UP as a command. Note that the action has to be specific:

```
try eating something;
```

is not allowed, since it doesn't say exactly what is to be eaten.

The word "try" is intended to make clear that there is no guarantee of success. For example:

```
Before locking the front door, try closing the front door.
```

could go wrong in any number of ways - perhaps the door is closed already, perhaps it is not openable, perhaps somebody has wedged it open. It would be safer to write:

```
Before locking the front door:
    try closing the front door;
    if the front door is open, stop the action.
```

There's no need to say anything if closing didn't work, because the closing action will have done that already. A neater approach still is to use:

silently try (action)

or:
try silently (action)

This phrase makes the action, which has to be named literally, take effect now, under the "silent" convention which means that routine messages aren't printed. Example:

```
try silently taking the napkin;
```

Silence is maintained only if this new action, the taking of the napkin, is successful (so if the napkin is successfully taken, the text "Taken." will not appear): if the action should fail, a suitable objection will be voiced as usual.

So now we have:

```
Before locking the front door:
    try silently closing the front door;
    if the front door is open, stop the action.
```

And this is neater because it won't produce a pointless "You close the front door." message.

See **Stored actions** *for how to store up actions as values and try those, too, so that isn't necessary to name the action as literally as in the examples above*

Start of Chapter 7: Basic Actions

Back to §7.3. Before rules

Onward to §7.5. After rules

Example 88: **Fine Laid** Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

Example 89: **Hayseed** A refinement of our staircase kind which can be climbed.

---

§7.5. After rules

There is pleasantly little to be said about "after" rules. If an action has survived all the rules in its way, and has actually succeeded, then we need to give the player a response which acknowledges this. Inform's normal rules will be sufficient to say something undramatic: for instance, if "taking the napkin" has succeeded then it will reply "Taken." to the player.

An after rule is an opportunity to say something more interesting:

```
After taking the diamonds, say "Taken!"
```

(Well, slightly more interesting.) After rules automatically end the action (as a success), which is what we would want in the above case. Allowing it to continue would simply result in "Taken." being printed as well. However, should we really need to do something and then carry on:

```
After taking the diamonds: say "(Mr Beebe looks up sharply.) ";
continue the action.
```

Start of Chapter 7: Basic Actions

Back to §7.4. Try and try silently

Onward to §7.6. Reading and talking

Example 90: ★ **Morning After** When the player picks something up which he hasn't already examined, the object is described.

§7.6. Reading and talking

A few actions apply not to items alone, but also involve what might be called conversation. The first is the one used for looking things up in books (which is conversation of a kind, even if the author is not present): "consulting ... about ...". For example,

```
In the Grove is a book of sybilline verses.

After consulting the book about "grove", say "The Grove is a sacred
yadda, yadda. There's a tree, that sort of thing. Wisdom."

After consulting the book about "future events", say "It's a bit, what's
```

```
the word? Delphic."
```

Note that what follows "about" here is a piece of text in double-quotes, and not the name of something. It can be almost any text at all, and in fact we shall later see (in the chapter on "Understanding") that we can match complicated patterns of words, too.

Similar actions are used for conversing with people:

```
After asking the Sybil about "verses", say "She blushes."

After telling the Sybil about "persians", say "She nods gravely."

After answering the Sybil that "I am mad", say "She sighs."
```

These would be produced by commands like "ask sybil about verses", "tell sybil about persians" and "answer i am mad". Answering is little-used except that it also catches commands like "sybil, something unrecognized", which inexperienced players sometimes type. Asking and telling, however, are important actions and the difference between them is often worth preserving. If you would prefer to make "tell sybil about X" do the same as "ask sybil about X", the following rule would serve:

```
Instead of telling the Sybil about something, try asking the Sybil
about it.
```

Games with a lot of conversation often involve great heaps of rules like the ones above, which can be repetitious to type out. We shall also later see (in the chapter on "Tables") that we can tabulate questions and answers in a much more concise way, if we prefer.

See **Topic columns** *for table-based ways to store and retrieve conversation*

Start of Chapter 7: Basic Actions

Back to §7.5. After rules

Onward to §7.7. The other four senses

Example 91: **Sybil 1** Direct

all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

Example 92: ★ **Lucy** Redirecting a question about one topic to ask about another.

Example 93: ★★ **Sybil 2** Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

Example 94: ★★★ **Costa Rican Ornithology** A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

---

§7.7. The other four senses

The five senses are all simulated with actions. Sight is so informative that it is handled by a whole range of actions: "looking", which describes the general scene; "examining something", which takes a closer look at a specific thing; "looking under something", and so on.

The other senses have one action each: "listening to something", "touching something", "tasting something" and "smelling something". It makes no sense to touch or taste the general scene, but listening and smelling are a different matter: we often just listen, without listening to anything specific. If the player types the command "listen", Inform understands that as listening to the current location: similarly for the bare command "smell". Thus:

```
Instead of listening to the Seashore, say "The song of gulls."
```

```
Instead of smelling the Cave, say "Salt and old seaweed."
```

Start of Chapter 7: Basic Actions

Back to §7.6. Reading and talking

Onward to §7.8. Rules applying to more than one action

Example 95: ★★★ **The Art of Noise** Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

§7.8. Rules applying to more than one action

A description can include more than one choice of action. For instance:

```
examining or searching the desk
```

matches either of "examining the desk" or "searching the desk". We can have more than two actions, of course:

```
examining, looking under or searching the desk
```

The actions combined like this need to be compatible with each other, at least a little. For instance, this will generate a problem message:

```
waiting or searching the desk
```

because it makes no sense to "wait the desk". On the other hand, this is fine:

```
waiting or searching
```

The general rule is that if we specify one or more objects ("the desk" in the above example), then each of the actions we quote must take at least that many objects.

For example, the following saves us writing the same basic rule three times over:

```
Instead of examining, looking under or searching the desk: say
"There's no use poking around in that old desk."
```

Start of Chapter 7: Basic Actions
Back to §7.7. The other four senses
Onward to §7.9. All actions and exceptional actions

§7.9. All actions and exceptional actions

The special description "doing something" (or "doing anything") matches any action, and "doing something to ..." also allows the noun to be specified.

For instance, the following puts its object out of bounds:

```
Instead of doing something to the cucumber sandwich, say "Lady
Bracknell stares disapprovingly down her pince-nez at you, in a way
which no amount of hunger or curiosity could overcome."
```

We sometimes need to be a little careful here: "waiting" qualifies as "doing something", but not as "doing something to something", because there is no object. "Putting the handbag on the cucumber sandwich" would also not qualify as "doing something to the cucumber sandwich" - only to the handbag.

More often, we would like to restrict the range of allowable actions to a select few. For instance:

```
Instead of doing something other than looking, examining or waiting:
say "You must learn patience."
```

(Or we can write "except" instead of "other than".) Or we might have an object, too:

```
Instead of doing something other than examining, taking or dropping
with the dagger: say "Don't fool around with that dagger. It's
exceedingly sharp."
```

Note the "with", which is crucial here. Without it, the rule is subtly different:

```
Instead of doing something other than examining, taking or dropping
the dagger: say "Don't fool around with that dagger. It's exceedingly
sharp."
```

This second version matches if the action is, say, taking a shield, or even just looking, because that would be an action other than examining the dagger, taking the dagger or dropping the dagger.

Start of Chapter 7: Basic Actions

Back to §7.8. Rules applying to more than one action

Onward to §7.10. The noun and the second noun

Example 96: ★ **Zodiac** Several variations on "doing something other than...", demonstrating different degrees of restriction.

---

§7.10. The noun and the second noun

Once we begin applying rules to actions which are not entirely known in advance, we have a problem: there's no way to find out what specifically is happening. Consider the following:

```
Instead of examining something, say "It is none of your concern!"
```

This is fine as far as it goes, but clumsy. What if the player had examined a human being? Then "it" would be inappropriate. A better approach would be this:

```
Instead of examining something, say "[The noun] is none of your
concern!"
```

The "noun" and, when necessary, the "second noun" are values which can be used in any rule about actions, and it follows that they can also be substituted into text, as this example demonstrates. Results might include:

```
Lady Bracknell is none of your concern!
```

```
The silver cigarette case is none of your concern!
```

This seems a good moment to mention that if you use "The" in a substitution, then a capitalised "The" will be used so long as this is grammatically correct (Lady Bracknell, as a proper noun, takes no article); "the" becomes a lower-case "the" along the same lines; and "a" a lower-case indefinite article.

```
Instead of examining something in the Drawing Room, say "Under
Lady Bracknell's eye, you feel constrained. Besides, it is only [a
noun]."
```

Start of Chapter 7: Basic Actions

Back to §7.9. All actions and exceptional actions

Onward to §7.11. In rooms and regions

Example 97: ★ **Ming Vase** ATTACK or DROP break and remove fragile items from play.

§7.11. In rooms and regions

Three elaborations of action descriptions increase the range of possibilities further.

```
Instead of taking something in the Supernatural Void, say "In this
peculiar mist you feel unable to grasp anything."
```

Like the objects to which the action applies, this location - the "in" clause - can take any description, not just an explicit place like "Supernatural Void":

```
Instead of listening in a dead end, say "You strain to hear further
clues as to the course of the underground river, but to no avail."
```

But we often want a rule to apply in any of a set of rooms: and where, unlike the "dead end" example above, the rooms have nothing much in common except where they happen to lie on a map. For instance, we might want a rule to apply only inside a given building, or a garden consisting of five miscellaneous rooms. If so, we can create a "region" as a convenient way to refer to that group of rooms:

```
The Arboretum is east of the Botanical Gardens. Northwest of the
Gardens is the Tropical Greenhouse.
```

```
The Public Area is a region. The Arboretum and Gardens are in the
Public Area.
```

```
Instead of eating in the Public Area, say "The curators of the Gardens
are ever among you, eagle-eyed and generally cussed."
```

Start of Chapter 7: Basic Actions
Back to §7.10. The noun and the second noun
Onward to §7.12. In the presence of, and when

§7.12. In the presence of, and when

Relative location can also be important: relative to other people, that is -

```
Instead of eating something in the presence of Lady Bracknell, say
"Lady Bracknell disapproves thoroughly of gentlemen who snack
between meals, and there are few disapprovals in this world quite so
thorough as Lady Bracknell's."
```

As might be guessed, this applies when the action takes place in the same location

as the person named: and of course that person can also be described more vaguely ("... in the presence of a woman", say), and can just as easily be an inanimate thing ("... in the presence of the radio set").

Lady Bracknell is a pushover compared to some matriarchs:

> Instead of doing something other than looking, examining or waiting in the presence of the Queen: say "I'm afraid they take what you might call a zero tolerance approach to breaches of court etiquette here."; end the story saying "You have been summarily beheaded".

The last of the optional clauses we can tack on to the description of an action is the most general of all. We can add "when" and then any condition at all, as in:

> Instead of eating something when the radio set is switched on, say "Something about the howling short-wave static puts you right off luncheon."

This supposes that the radio is so loud that it can be heard from any room: we could muffle it so that it's only audible from the room it is in like so:

> Instead of eating something in the presence of the radio set when the radio set is switched on, say "Something about the howling short-wave static puts you right off luncheon."

Start of Chapter 7: Basic Actions

Back to §7.11. In rooms and regions

Onward to §7.13. Going from, going to

Example 98: ★ **Beachfront** An item that the player can't interact with until he has found it by searching the scenery.

Example 99: ★★ **Today Tomorrow** A few notes on "In the presence of" and how it interacts with concealed objects.

---

§7.13. Going from, going to

Going is an action defined like any other: it is the one which happens when the player tries to go from one location to another. But it is unlike other actions because it happens in two locations, not just one, and has other complications such as vehicles and doors to contend with. To make it easier to write legible and flexible rules, "going" is allowed to be described in a number of special ways not open to other actions, as demonstrated by the following example story:

```
"Going Going"

The Catalogue Room is east of the Front Stacks. South of the
Catalogue Room is the Musicology Section.

Instead of going nowhere from the Front Stacks, say "Bookcases
obstruct almost all passages out of here."

Instead of going nowhere, say "You really can't wander around at
random in the Library."

Before going to the Catalogue Room, say "You emerge back into the
Catalogue Room."
```

Note that "going nowhere" means trying a map connection which is blank, and if no rules intervene then "You can't go that way" is normally printed. Unless "nowhere" is specified, descriptions of going apply only when there is a map connection. So "going from the Musicology Section" would not match if the player were trying to go east from there, since there is no map connection to the east. Similarly, "going somewhere" excludes blank connections.

The places gone "from" or "to" can be specific named regions instead of rooms. This is convenient when there are several different ways into or out of an area of map but a common rule needs to apply to all: so, for example,

```
Before going from the Cultivated Land to the Wilderness, ...
Before going nowhere from the Wilderness, say "Tangled brush
forces you back."
```

Note that it must be "going nowhere from the Wilderness", not "...in the Wilderness". (Note also the caveat that the regions must be named: "going from a region", or something similarly nonspecific, will not work.)

An important point about "going... from" is that, as mentioned in general terms above, it requires that there is actually a map connection that way: whereas "going... in" does not. Suppose there is no map connection north from the Wilderness. Then:

```
Instead of going north from the Wilderness, say "You'll never read
this."
Instead of going north in the Wilderness, say "Oh, it's too cold."
```

The first of these never happens, because it is logically impossible to go north from the Wilderness: but the second does happen. (Technically, this is because "going north" is the action, and "in the Wilderness" a separate condition tacked onto the rule.) This distinction is often useful - it allows us to write rules which apply only to feasible movements.

This may be a good place to mention a small restriction on the ways we can specify an action for a rule to apply to, and how it can be overcome. The restriction is that the action should only involve constant quantities, so that the following does not work:

```
The Dome is a room. The Hutch is north of the Dome. The rabbit is in
the Hutch. Before going to the location of the rabbit, say "You pick up
a scent!"
```

because "the location of the rabbit" is a quantity which changes in play (the player can pick up the rabbit and take him to the Dome, for instance). However, we can get around this restriction by defining a suitable adjective, like so:

```
The Dome is a room. The Hutch is north of the Dome. The rabbit is in
the Hutch. Definition: a room is rabbit-infested if it is the location of
the rabbit. Before going to a rabbit-infested room, say "You pick up a
scent!"
```

Start of Chapter 7: Basic Actions
Back to §7.12. In the presence of,

and when

Onward to §7.14. Going by, going through, going with

Example 100: ★ **Veronica** An effect that occurs only when the player leaves a region entirely.

Example 101: ★★ **A&E** Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

Example 102: ★★★ **Bumping into Walls** Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

Example 103: ★★★ **Polarity** A "go back" command that keeps track of the direction from which the player came, and sends him back.

---

§7.14. Going by, going through, going with

Adding to the previous example story, we apply rules which depend on travelling by a particular vehicle:

> The book trolley is in the Musicology Section. "The book trolley, a sort of motorised tractor for trundling around through the stacks, is parked here." The trolley is a vehicle. Instead of going nowhere by the trolley, say "Don't go crashing the trolley into walls."

> Instead of going to the Front Stacks by the trolley, say "The Front Stacks are far too confined for the trolley to manoeuvre into them."

And, lastly, rules which apply to movements through particular doors:

```
The green baize door is east of the Catalogue Room and west of the
Clerk's Office. The green baize door is an open door.
```

```
Before going through the green baize door, say "Through you go..."
After going through the green baize door: try looking; say "...and here
you are."
```

(Note that these apply whether the action is "going east" or "entering the green baize door", each having the same effect.) The last rule is worth a second look: the normal way that a "going" action is reported is to produce the room description of the new location. So if an "after" rule stops the action before we get to reporting, we have to produce any room description by hand (hence the "try looking" to cause the looking action). Alternatively, we could simply say something and let the normal course of events take place:

```
After going through the green baize door: say "...and here you are:";
continue the action.
```

Finally, going is an action which can also happen while the player is pushing something from one room to another, and we can describe this like so:

```
Instead of going from the Office with the trolley, say "But it looks
perfectly placed here. Why push any further?"
```

"Going" is not the only action which moves the player. Another is "exiting", an action which moves the player out of whatever he/she is currently in or on. This action is often caused by the player typing just OUT or GET DOWN, and there's no noun as such. But Inform allows the syntax "exiting from" to make it easier to write rules about the exiting of particular containers or supporters:

```
After exiting from the Mini Cooper:
    say "You painstakingly unpack your limbs from the tiny car."
```

Start of Chapter 7: Basic Actions

Back to §7.13. Going from, going to

Onward to §7.15. Kinds of action

Example 104: ★ **No Relation** A car which must be turned on before it can be driven, and can only go to roads.

Example 105: ★ **Mattress King** Adding extra phrasing to the action to PUSH something in a direction.

Example 106: ★★ **One Short Plank** A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

Example 107: ★★★ **Provenance Unknown** Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

Example 108: ★★★ **Zorb** Replacing the message the player receives when attempting to push something that isn't pushable, and also to remove the restriction that objects cannot be pushed up or down.

---

§7.15. Kinds of action

Especially when people need to react to events going on around them, it is helpful to be able to categorise actions into whole areas of behaviour. For instance:

Kissing Mr Carr is unmaidenly behaviour.
Doing something to the painting is unmaidenly behaviour.

```
Instead of unmaidenly behaviour in the Inn, say "How unmaidenly!"
```

Here a new kind of action called "unmaidenly behaviour" has been created and then used in the description of an instead rule. The convenience of this approach is that when further actions suddenly occur to us as also being unmaidenly - say, attacking Mr Carr - we only need to add a single line:

```
Attacking Mr Carr is unmaidenly behaviour.
```

And this will automatically be reflected in any rules which concern the consequences of failing to be ladylike.

(Note that we were only allowed to say that "Kissing Mr Carr is unmaidenly behaviour." because Inform already knew from earlier sentences - see the example below - that Mr Carr was a person, and therefore that "kissing Mr Carr" made sense as a description of an action.)

Start of Chapter 7: Basic Actions

Back to §7.14. Going by, going through, going with

Onward to §7.16. Repeated actions

Example 109: ★ **Dearth and the Maiden** Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and seemliness.

Example 110: ★★★ **Mimicry** People who must be greeted before conversation can begin.

---

§7.16. Repeated actions

We come at last to the final thing which can be specified about an action, and appropriately enough it must be specified with the final words of the description. This is the way to talk about repeated activity:

Instead of examining the tapestry for the third time, say "All right, so it's a masterpiece, but is this really the time to make a detailed study?"

Instead of examining the urn at least twice, say "It's an urn. What do you want from me?"

Instead of going nowhere for the 20th time, say "Do stop walking into walls, there's a good fellow."

Note that we are allowed to spell out numbers up to twelve in English words, but beyond that must use digits (thus "twelfth" is allowed but not "thirteenth": "13th" should be used instead). The following example is instructive:

Instead of taking something for the fourth time, say "No. I'm capricious."

This means that it is the fourth time a "taking..." action has been tried, and does not mean that the same item was taken each time. Also, note that we are counting the number of times the action has been tried, not the number of times it succeeded.

Start of Chapter 7: Basic Actions

Back to §7.15. Kinds of action

Onward to §7.17. Actions on consecutive turns

Example 111: ★ **Y ask Y?** Noticing when the player seems to be at a loss, and recommending the use of hints.

Example 112: ★★★★ **A Day For Fresh Sushi** A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is

§7.17. Actions on consecutive turns

We can also reckon the number of consecutive turns on which an action has been repeated, by talking about "turns" instead of "times", as demonstrated in the following example story. Note also that we are allowed to use the phrase "doing it" to mean "the same description as the previous one except for the part about turns or times", like so:

```
"Waiting Room"

The Antechamber is a room. The tattered copy of Women's Wear
Daily is in the Antechamber. Instead of taking the Daily, say "It is
stamped NOT TO BE TAKEN AWAY."

Instead of examining the Daily for the first time, say "The best article
seems to be about how your star sign affects your best swimsuit
colour. Really: that's the best article."

Instead of doing it for the second time, say "You now know a
generous amount about a typical week in the life of a weather
forecaster."

Instead of doing it for the third time, say "You would now know how to
cook herb bread, except that you have already forgotten the names
of both of the herbs."

Instead of doing it more than three times, say "Nope, you've drained
it of all conceivable sustenance, even the small ads about French
farmhouses to let (sleeps 7) and breast reduction surgery (with
alarming photographs in sallow light)."
```

```
After waiting for four to six turns, say "This is getting mighty dull."
After waiting for seven to eight turns, say "Really, exceptionally dull."
After waiting for nine turns, end the story saying "You have died of
boredom, something previously thought medically impossible".
```

Note once again that numbers above twelve must not be written out, so "more than twelve times" would be acceptable, but we would write "more than 13 times".

Start of Chapter 7: Basic Actions
Back to §7.16. Repeated actions
Onward to §7.18. Postscript on actions

§7.18. Postscript on actions

In this chapter, all actions have been carried out by the player, all have been drawn from the standard stock of built-in actions ("unlocking", "taking", "going" and so forth), and all of those built-in actions have been allowed to work in the standard way - we have seen how to prevent the taking of something, and how to give this unexpected consequences, but not how to make taking work in an entirely different way.

All three of those restrictions will later be lifted in the chapter on "Advanced Actions", but otherwise we have covered the ground thoroughly, and it is time to move on to the techniques enabling us to do more than make tart replies to the player: it is time to change the world.

Start of Chapter 7: Basic Actions
Back to §7.17. Actions on consecutive turns
Onward to Chapter 8: Change: §8.1. Change of values that vary

Examples from Chapter 7: Basic Actions

Start of this chapter
Chapter 8: Change
Indexes of the examples

ExampleGrilling

A grill, from which the player is not allowed to take anything lest he burn himself.

RB

Descriptions of objects can be used in "Instead" rules: we can not only say "Instead of taking the steak", but also "Instead of taking something" or "Instead of taking something which is on the grill".

That last rule is useful if, for example, we want to prevent the player from interacting with anything on a specific supporter:

```
"Grilling"

The Patio is a room. The Patio contains a grill and an
ice chest. The ice chest contains a cold beer. On the
grill are a steak and a hot dog.

Mom is a woman in the Patio.

Instead of taking something which is on the grill:
    say "'Hey, you'll burn yourself,' says Mom."

Test me with "get steak / get all from grill / get all".
```

We could just as easily adapt this rule to affect a container: "Instead of taking something which is in the ice chest," for example.

Note also that in older versions of Inform, the pattern "get all from..." was treated differently from "get steak", and had to be accounted for separately. This is no longer the case; this instead of taking... rule will handle all the phrasings which the player might use to try to acquire this object.

Change the player's appearance in response to EXAMINE ME.

85 *RB*

```
"Bad Hair Day"

The Foyer is a room. "A mirror hangs over the table,
tempting you to check your appearance before going in
with all the others."

Instead of examining the player:
    say "Oh, stop fussing. You look fine."

Test me with "examine me".
```

Example **Democratic Process**

86 *RB*

Make PUT and INSERT commands automatically take objects if the player is not holding them.

"Stop" and "Continue" are most useful when we need to write rules that will have to stop the action some of the time but at other times let it pass; so for instance:

```
"Democratic Process"

Before inserting something which is not carried by the
player into something:
    if the noun is in the second noun, say "Already done."
instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
```

```
    if the player is not holding the noun, stop the action.

Before putting something which is not carried by the
player on something:
    if the noun is on the second noun, say "Already
done." instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

The Assembly Room is a room. "On most days, this
room is used for elementary school assemblies; at the
moment, it serves as a voting place." The ballot is on
the desk. The desk is in the Assembly Room.

The machine is a container in the Assembly Room. "On
the ballot machine is a sign which reads 'PUT BALLOTS
IN ME :)'." Understand "ballot machine" as the machine.

Test me with "put ballot in machine".
```

---

87. ExampleSand *RB*

Extend PUT and INSERT handling to cases where multiple objects are intended at once.

The above example does not quite work when we want the player to be allowed to take multiple objects at once before putting them somewhere: we also need to add a couple of "understand" rules borrowed from many chapters later. While the reasons may not be immediately clear, we include the demonstration here for the sake of thoroughness:

```
"Sand"
```

Before inserting something which is not carried by the player into something:
    if the noun is in the second noun, say "Already done." instead;
    say "(first taking [the noun]) ";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

Before putting something which is not carried by the player on something:
    if the noun is on the second noun, say "Already done." instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

Understand "put [things] in [something]" as inserting it into. Understand "put [things] on [something]" as putting it on.

The Closet is a room.

A lentil is a kind of thing. A black-eyed pea is a kind of thing. The closet contains 3 lentils. The Closet contains 14 black-eyed peas. The round tin is a container in the closet. The round tin contains 17 lentils. The square tin is a container in the Closet. The square tin contains 20 black-eyed peas.

Sorting is a scene. Sorting begins when play begins. Sorting ends when all the lentils are in the round tin and all the black-eyed peas are in the square tin. When Sorting ends, end the story finally.

When play begins: say "Thanks to your cruel stepmother, you're not going anywhere until the lentils

and peas are sorted."

Test me with "put peas in square tin / put lentils in round tin".

Example **Fine Laid** *RB*

Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

Sometimes it is useful to direct all -- or almost all -- actions from one object to another. For the sake of argument, say we have a sheet of paper with writing on it, and (because we're very meticulous) we want to let the player examine the writing and get a customized response, different from when he just examines the sheet of paper. But for all other purposes -- say, TAKE or TASTE -- we want the two objects to be treated as one.

Here, we approach the problem by changing the noun and/or the second noun of the current action, then issuing a new command to "try the current action". Because we've changed the noun and second noun, the "current action" at this point is different from the one generated originally by the player's command.

"Fine Laid"

High Street Stationer is a room.

The sheet of paper is a thing in High Street Stationer. The writing is part of the sheet of paper.

The description of the sheet of paper is "A beautiful sheet of heavy cream paper." The description of the writing is "Delicate and spidery."

Instead of tasting the sheet of paper, say "You might need more fiber in your diet, but this isn't the way.".

Before doing something other than examining when the current action involves the writing:
    if the writing is the noun, now the noun is the sheet of paper;
    if the writing is the second noun, now the second noun is the sheet of paper;
    try the current action instead.

Test me with "examine sheet of paper / examine writing / get writing / taste writing".

---

89 ExampleHayseed *RB*

A refinement of our staircase kind which can be climbed.

Presumably all staircase-type connections between rooms should respond when the player says CLIMB STAIRS (or the equivalent). So

"Hayseed"

A staircase is a kind of door. A staircase is usually open. A staircase is seldom openable.

The ladder is a staircase. It is above the Barn and below the Hayloft.

Instead of climbing a staircase:

```
try entering the noun.


Test me with "climb ladder / g".
```

Attempts to climb other types of door will still be treated as useless.

Example **Morning After** *RB*

When the player picks something up which he hasn't already examined, the object is described.

Suppose we want to make the player's life slightly easier by examining everything he picks up, if he hasn't already examined it.

```
"Morning After"


A thing can be examined or unexamined.


After taking something unexamined:
    say "Taken. [run paragraph on]";
    try examining the noun.


Carry out examining something:
    now the noun is examined.
```

Carry out rules are explained in more detail in the chapter on advanced action handling. For now, it may be enough to know that what we put into this carry out rule for examining will happen any time anything is examined, but that it will not interfere with the rest of the predefined behavior of the action. The player will still see the object description and so on, as usual.

The Red Door Saloon is a room. "This old place is in pretty bad shape since the mine shut down. Now there's not much to see but the pair of deep gouges in the floorboards where they dragged away the Sheriff's corpse with the spurs still on."

Jed is a man in the Red Door Saloon. "At 8:30 AM the only person around is old Jed, collecting his hangover cure."

The pistol is a thing in the Red Door Saloon. The description of the pistol is "It ain't too accurate, but for two dollars you can't expect much."

The hangover cure is a thing in the Red Door Saloon. The description of the hangover cure is "Two yellow egg-yolks unbroken in a red-brown liquid. Yep."

Test me with "x pistol / get all".

---

g1 ExampleSybil 1 *RB*

Direct all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

Sometimes we do not particularly want to deal with all the variations on asking, telling, or answering someone something, but want to direct everything to a single conversational command:

"Consulting the Oracle"

The Grove is a room. In the Grove is a woman called

```
the Sybil.
```

```
Instead of telling someone about something, try asking
the noun about it. Instead of answering the noun that
something, try asking the noun about it.
```

```
Instead of asking the Sybil about "persians", say "She
nods gravely."
```

And similarly, a difference between GIVE and SHOW is sometimes overkill:

```
Instead of showing something to someone, try giving the
noun to the second noun.
```

```
The player carries a coin. Instead of giving the coin to
the Sybil: move the coin to the Sybil; say "She accepts
with a smile."
```

It is also often the case that we want to accept more than one form of a term. For instance

```
Instead of asking the Sybil about "Darius/king", say "Her
smile unnerves you."
```

will match either "Darius" or "king". If necessary, we can go a step further and define our own token to match a variety of phrases, like this:

```
Understand "Athenians/Spartans/Greeks" or "hoplite
army/forces" as "[Greeks]". Instead of asking the Sybil
about "[Greeks]", say "She looks encouraging."
```

The token "[Greeks]" will match all of "Athenians", "Spartans", "Greeks", "hoplite army", or "hoplite forces". It will not match "hoplite" or "forces" alone; it is important to note that the / divides individual words which are understood equivalently, but does not define entire phrases as

equivalent. More about how Inform understands specific phrases can be found in the chapter on Understanding.

```
Test me with "test one / test two".

Test one with "ask sybil about persians / tell sybil about
persians / sybil, persians / ask sybil about darius / ask
sybil about king".

Test two with "ask sybil about greeks / ask sybil about
athenians / ask sybil about hoplite army / ask sybil about
hoplite forces / give the coin to the sybil".
```

---

g2 ExampleLucy

RB

Redirecting a question about one topic to ask about another.

Occasionally we will want to replace the player's question topic with another of our own devising. We can do this in the simplest possible case like so:

```
"Lucy"

The International Boardgame Championship is a room.
Lucy is a woman in the Championship.

Instead of asking Lucy about "checkers":
    try asking Lucy about "games".

Instead of asking Lucy about "games",
```

```
say "'I don't like games,' she sniffs."
```

```
Test me with "ask lucy about checkers / ask lucy about
games".
```

Note that this syntax did not work in older versions of Inform; it is now safe.

ExampleSybil 2

93 *RB*

Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

Inform already understands YES, NO, and SORRY as commands in their own right, which can make things a little sticky when we want a character to ask a question of the player. The most important thing is not to cover some of the possible phrasings while ignoring others.

```
"Replies"
```

```
The Grove is a room. In the Grove is a woman called
the Sybil.
```

```
Instead of asking the Sybil to try saying no: try saying
no. Instead of asking the Sybil to try saying yes: try
saying yes. Instead of asking the Sybil to try saying
sorry: try saying sorry.
```

```
Instead of answering the Sybil that "yes", try saying yes.
Instead of answering the Sybil that "no", try saying no.
Instead of answering the Sybil that "sorry", try saying
sorry.
```

```
Instead of saying yes in the presence of the Sybil:
    say "She looks interested."

Instead of saying no in the presence of the Sybil:
    say "She looks annoyed."

Instead of saying sorry in the presence of the Sybil:
    say "She looks bored."
```

The complexity arises from the fact that we want to handle both YES and SYBIL, YES. If we only had the latter, 'yes' would be treated as a text given to the Sybil, just as in the commands SAY YES TO SYBIL or ANSWER YES. But because we have defined it as a command (so that the player can use it independently), SYBIL, YES is understood as an order to the Sybil to do the YES action.

Fortunately, we can redirect everything, as here, so that the results wind up the same.

And if we want yet another variation not covered by the Inform standard:

```
Understand "tell [someone] [text]" as answering it that.
Understand "tell [someone] that [text]" as answering it
that.
```

But that is a matter for a later chapter.

```
Test me with "yes / sybil, yes / say yes to sybil / answer
yes / tell sybil yes / no / sybil, no / say no to sybil /
answer no / tell sybil no / sorry / sybil, sorry / say sorry
to sybil / answer sorry / tell sybil sorry".
```

Example **Costa Rican Ornithology** *RB*

A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

The following relies on quite a number of features we haven't met yet: tables, rules for printing names, instructions for understanding the player's commands. It is offered simply as an example of how a fully implemented book might be handled in Inform.

```
"Costa Rican Ornithology"

A book is a kind of thing. Understand "book" as a book.
A book has a table name called the contents.

Instead of consulting a book about a topic listed in the
contents of the noun:
    say "[reply entry][paragraph break]".

Report consulting a book about:
    say "You flip through [the noun], but find no reference
to [the topic understood]." instead.
```

With this "topic understood" phrase, we're telling Inform to print back the word or phrase that the player was attempting to look up. This overrides the more general default response, "You discover nothing of interest in the book."

We now have the essential elements to construct whatever books we like. Now let's have an example of a specific book:

```
The Guide to Central American Birds is a book carried
by the player. The contents of the Guide is the Table of
Listed Birds.
```

We will come back to the idea of tables and table names later, but for now the important thing is that we have instructed Inform to look up its answers to consulting the bird guide in this form:

```
Table of Listed Birds
```

topic
reply
"[red]" or "red bird" or "red"
"[red]" or "red bird" or "red" for a while and eventually discover a reference to the [scarlet macaw], which appears to correspond with what you see before you."
"The quetzal is quite regal, describing its brilliant plumage, flashing and igniting in the sunshine, which is supposedly sufficient to lure birdwatchers from all over the world. Unfortunately, the quetzal is described as being bright emerald in color, with a pink fuzz on its head and a long soft tail 'like a feather boa'. None of these describes your visitor."

The topic column is a bit special: it matches the player's input, and is not meant to be printed out again. Topic columns will be discussed further in the chapter on Tables. (Note also that, however it may appear in the documentation, the topic column should not be spanning multiple lines in our source text.)

We may also compress long or complicated topics by creating bracketed abbreviations, and in fact it's useful to do so now, to explain the red token we just used:

```
Understand "red-orange" or "bird" or "red" or "orange"
as the scarlet macaw. Understand "red-orange" or "red"
or "orange" or "scarlet" as "[red]".
```

This technique is discussed further in the chapter on Understanding.

If we wanted more books, we could define those in the same way, giving each its own separate contents table to be used for consultation. But for the sake of the example we will keep it simple, and move on to the scenario itself:

```
The Veranda is a room. "From here you can see a
considerable expanse of dense-growing jungle plants,
and eventually the open water beyond."


The scarlet macaw is an animal in the veranda. "A
vibrantly-colored [scarlet macaw] perches on the rail."


A thing can be known or unknown.
```

```
Before printing the name of the scarlet macaw while
consulting:
    now the scarlet macaw is known.


Rule for printing the name of the unknown scarlet
macaw: if the macaw is unknown, say "red-orange bird
of unknown species".


Test me with "look up penguins in the guide / look up
quetzal in guide / look up silver nuthatches in the guide /
look / look up red bird in the book / look".
```

Example **The Art of Noise** *RB*

Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

This example involves redesigning the LISTEN command, removing its built-in function and replacing that with something more ambitious. We will learn more about how to do this later on.

```
"The Art of Noise"


A thing has some text called sound. The sound of a
thing is usually "silence".


The report listening rule is not listed in the report
listening to rules.


Carry out listening to something:
```

```
    say "From [the noun] you hear [the sound of the
noun]."


Instead of listening to a room:
    if an audible thing can be touched by the player, say
"You hear [the list of audible things which can be
touched by the player].";
    otherwise say "Nothing of note."


Definition: a thing is audible if the sound of it is not
"silence".


Before printing the name of something audible while
listening to a room:
    say "[sound] from the "


The Sharper Image is a room. The pet rock is a thing in
the Sharper Image. The toy car is a thing in the Sharper
Image. The sound of the car is "whirring and zooming".
The plastic widget is a thing in the Sharper Image. The
sound of the plastic widget is "bleeps and bloops". The
pointless gadget is a thing in Sharper Image. The sound
of the pointless gadget is "buzzbuzzbuzz".


The soundproof case is a transparent openable
container in the Sharper Image. It is closed and fixed in
place.


Test me with "listen / listen to rock / listen to car / get all
/ open case / put all in case / listen / close case / listen /
listen to car".
```

Several variations on "doing something other than...", demonstrating different degrees of restriction.

Notice that the following two scenarios do not have the same effect:

```
"Zodiac"

The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.

The player carries a can of mace and a roll of duct tape.

Instead of doing something other than examining with
the Capricorn Killer:
    say "You wouldn't dare!"

Test me with "x me / x killer / touch killer / smell mace".
```

...will prevent the player from doing anything else to the killer, but allow him free range of action with other objects. By contrast, the following will prevent him doing anything other than examining to any item:

```
"Zodiac"

The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.

The player carries a can of mace and a roll of duct tape.
```

```
Instead of doing something other than examining
something in the presence of the Capricorn Killer:
    say "You dare not attempt it!"


Test me with "x me / x killer / touch killer / smell mace".
```

But notice that because we specified "examining something", actions that take no object (like sing) are still not affected.

The most strict statement would be

```
"Zodiac"


The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.


The player carries a can of mace and a roll of duct tape.


Instead of doing something other than looking or
examining in the presence of the Capricorn Killer:
    say "You dare not attempt it!"


Test me with "x me / x killer / touch killer / smell mace".
```

Because we left "something" out, now the pattern does not have to match an activity with an object; it will match any activity at all. Singing too will be deemed too risky in the presence of our sinister foe. But we do need to make an exception for "look", or else we won't get even the room description.

Example Ming Vase

ATTACK or DROP break and remove fragile items from play.

RB

In emulation of a certain annoying aspect of the original Adventure, in which there is a Ming vase that cannot safely be dropped:

```
"Ming Vase"

A thing can be strong or fragile. A thing is usually
strong.

Instead of attacking or dropping a fragile thing:
    now the noun is nowhere;
    say "[The noun] breaks into thousands of pieces!"

The Cave is a room. The Ming vase is carried by the
player. The vase is fragile.
```

We could also implement an additional refinement from Adventure, that a fragile thing is safe if dropped when there is a cushion nearby.

```
The pillow is a portable supporter. It is carried by the
player.

Instead of dropping a fragile thing when the pillow is in
the location: try putting the noun on the pillow instead.

After putting a fragile thing on the pillow:
    say "You set [the noun] down gently on the pillow."
```

```
Test me with "drop pillow / drop vase / get vase / get
pillow / drop vase".
```

In this rule, the pillow is the second noun; if we had a general rule about setting fragile things on soft things, we could say "You set [the noun] down gently on [the second noun]." for the same effect.

If we wanted to be more refined, we would provide extra code so that breaking a container or a supporter would leave behind their contents. We will see how to do that later.

ExampleBeachfront

An item that the player can't interact with until he has found it by searching the scenery. *RB*

Suppose we have our player, a detective, searching for evidence; we don't want him to be able to use this evidence until he has performed the action that reveals it, but after that it should be visible in the room when he looks.

A simple way to do this is to start the object -- an envelope, in this scenario -- out of play, and only move it into the location when the player looks for it:

```
"Beachfront"

The Stuffy Office is a room. "The windows are closed,
making the sultry air even more unbearable. A narrow
slice of Caribbean blue is visible between the scuba
gear rental shop and the recreated 17th century pirate
tavern.

The office is cheerfully furnished with wicker chairs and
white curtains, but the tropical decorating scheme
stopped at the desk, which is heavy oak and absolutely
covered with papers."
```

```
The heavy oak desk is a supporter in the stuffy office. It
is scenery. Understand "paperwork" as the desk.
```

```
The creamy envelope is an openable container. The
description is "There is no return address on the outside
of the envelope, just the address of the Doctor's office --
but the legs of the capital A are rubbed down in a
characteristic way, and the top of every R is open.
There's no question that it comes from the same
typewriter as the blackmail note." In the envelope is a
letter. The envelope can be found or lost. The envelope
is lost.
```

```
Instead of searching the desk when the envelope is lost:
    now the envelope is found;
    say "You rifle through the piles of bills and notices;
invitations to conventions; advertisements for high-end
prescription drugs; pink carbon sheets bearing
patients['] names and medical identification numbers in
spidery, elderly handwriting. Almost at the bottom of the
heap, you find what you were looking for: a creamy
envelope with the address typed.";
    move the envelope to the desk.
```

Here we've changed the property of the envelope to keep track of the fact that it has been found, so that if the player tries again, he won't find anything more.

```
Instead of searching the desk:
    say "Further investigation of the desk reveals nothing
else suspicious."
```

Notice that we have two rules that apply to "searching the desk", but one of them has a more specific set of parameters ("when the envelope is lost"). This means that Inform will consult that rule first and use it if it applies; it will only carry out our plain vanilla "instead of searching the desk" rule when the more restricted rule is not relevant.

```
Test me with "x envelope / x desk / search desk / look /
get envelope / x envelope".
```

Rules about concealment will affect "in the presence of", too. For instance, suppose we have a man with a pocket pet:

"Today Tomorrow"

The Temporary Employment Office is a room. "'Tomorrow's Temporary Workers - Today!' proclaims the logo over the door. The office is divided into two areas, the inner sanctum where you take calls and fiddle with the computer, and the outer area where workers take skill exams and watch inspirational videos ('Earn your way to partial benefits!', 'Vacation days and you!', 'Temping the Tomorrow Way', etc.)."

Maya is a woman in the Office. "Your coworker Maya sits at her own computer, diligently modifying all the [one of]pay rates in the database down from $9.00 an hour to $8.96[or]billing rates in the database up from $25.00 an hour to $25.04[purely at random]." She wears a trenchcoat. She carries a chihuahua. The description of Maya is "[if Maya is wearing the trenchcoat]She's wearing a trenchcoat, in a bizarre bid to keep your boss unaware of the chihuahua in her pocket. 'Because petsitters are really expensive!' she explained to you in an urgent hiss this morning over the coffeemaker. [otherwise]She looks cold. [end if]To all appearances, she is carrying [a list of unconcealed things carried by Maya]."

Maya's computer is scenery in the Employment Office. The description is "You can't see the screen from here, but she was perfectly happy to tell you what she was doing."

Rule for deciding the concealed possessions of someone (called carrier):
    if the particular possession is the chihuahua and the carrier wears the trenchcoat, yes;
    otherwise no.

Instead of eating something in the presence of the chihuahua:
    say "[The chihuahua] yips at you! Maya looks despairingly at [the noun], which is obviously inciting it."

The desk is scenery in the Office. On the desk are a multi-line telephone and a printer. The description of the printer is "Every morning, this instrument of torture spits out a list of the assignments you have to fill - professional, attractive receptionist with level three Excel certification, at $7.05 an hour; that sort of thing. You spend the ensuing three hours trying to meet its tyrannical demands." The description of the multi-line telephone is "Line three blinks urgently at you. You'll pick it up again as soon as you can remember who it was you put on hold."

The player carries a double bacon cheeseburger and a chocolate shake. Understand "milkshake" or "beverage" as the shake. The description of the cheeseburger is "A minor self-indulgence to make up for the fact that you have to work through lunch." The cheeseburger is edible. The shake is edible. The description of the shake is "It has the consistency of spackle and is no longer cold, but there is some chocolatey goodness in it still." Understand "burger" or "hamburger" as the cheeseburger. Instead of drinking the chocolate shake: try eating the shake instead.

Test me with "x maya / x cheeseburger / drink shake / eat cheeseburger".

Unless we somehow get the trenchcoat away from Maya, the chihuahua

will not be in view, and will not intervene in our lunch. All very well for the player character, but not so interesting to the story... To this end, we might add an unfortunate event, courtesy of later chapters:

```
The time of day is 11:45 AM.


At 11:47 AM: say "Your boss pokes his head in,
temporarily free of the round of conference calls that
occupy all his days. 'Maya,' he says. 'Your coat?' He
shakes his head, clucking sadly. 'It doesn't say
professional!' But mercifully Maya manages to take it off
so slowly that he doesn't glimpse her pet before her
phone rings again.";
    now Maya carries the trenchcoat.
```

---

100 ExampleVeronica *RB*

An effect that occurs only when the player leaves a region entirely.

Suppose that we want to have something happen when the player leaves a region we've defined. "Instead of going from (the region)..." will not suffice for this, because this rule will be invoked every time the player successfully leaves a room within the region, whether or not he is going to a room that is also in the same region.

Instead we need a rule that is a bit more specific, like this:

```
"Veronica"


Neptune is a region.


Tijuana is a room.
```

```
High School is north of Tijuana. It is in Neptune.

Detective Offices is west of High School. It is in
Neptune.

The player is in High School.

Instead of going from Neptune to a room which is not in
Neptune:
    say "It's a bad time to leave Neptune."

Test me with "s / w / e".
```

---

101 ExampleA&E *RB*

Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

Rules about going to regions make it easy to exclude the player from a large portion of the map, even if there are many connecting paths to the region. For instance, in this story it would be annoying to have to write a rule about all four exits by which the player could reach the film set area:

```
"A&E"

Winding Street is a room. Winding Street is west of
Duck Pond. Sloping Street is north of Winding Street,
northwest of Duck Pond, west of Stately Lawn, and
southwest of Stately Home. Stately Lawn is north of
Duck Pond. Stately Home is north of Stately Lawn.
```

```
Film Set is a region. Duck Pond, Stately Lawn, and
Stately Home are in Film Set.


Instead of going to Film Set when the player does not
carry the VIP Pass: say "A burly studio guard
materializes in your path, convincing you that you would
prefer to be elsewhere."


The VIP Pass is in the garbage can. The garbage can is
in Sloping Street.


After going to the Film Set:
    say "Success! At last you are inside the set of
'Prouder and More Prejudiced'. Next step: locating Mr
Firth.";
    end the story finally.


Test me with "e / n / e / get pass / e".
```

Example Bumping into Walls *RB*

Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

```
"Bumping into Walls"
```

First we add an instruction to determine which ways lead to other rooms.

```
Definition: a direction (called thataway) is viable if the
room thataway from the location is a room.
```

Now we build in the instruction for what Inform should say if the player tries to head in a direction that leads nowhere:

```
Instead of going nowhere:
    let count of exits be the number of viable directions;
    if the count of exits is 0, say "You appear to be
trapped in here." instead;
    if the count of exits is 1, say "From here, the only way
out is [list of viable directions].";
    otherwise say "From here, the viable exits are [list of
viable directions]."
```

There is no theoretical reason why we have to define "count of exits" here: we could, if we wanted, just say "if the number of viable directions is 0", "if the number of viable directions is 1", and so on. However, each calculation of a "viable direction" takes a bit of computing power, so there is some slight savings in not requiring the game to count viable directions more than once in this routine.

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The Crypt is below the dome.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "u / n / n / e / n / s / u / open door / e / n".
```

A "go back" command that keeps track of the direction from which the player came, and sends him back.

The main trick of this is always to record where the player has gone when he has just moved.

```
"Polarity"

The former location is a room that varies.
```

Here we record where the player has been before moving him; by calling this the "first carry out going rule", we make sure that this rule is followed during the going action before any other pieces of the movement occur. For more detail, see the chapters on advanced actions and on rules.

```
First carry out going rule:
    now the former location is the location.

Understand "go back" as retreating. Understand "back"
or "return" or "retreat" as retreating.

Retreating is an action applying to nothing.

Carry out retreating:
    let way be the best route from the location to the
former location, using doors;
    if way is a direction, try going way;
    otherwise say "You can't see an open way back."
```

And to deal with the case where the player has not yet moved:

```
When play begins: now the former location is the Dome.
```

Instead of retreating when the former location is the location: say "You haven't gone anywhere yet."

Dome is a room. North of Dome is North Chapel. South of the Dome is South Chapel. West of the Dome is Western End. Quiet Corner is northwest of the Dome, north of Western End, and west of North Chapel. Loud Corner is east of North Chapel, northeast of Dome, and north of Eastern End. Eastern End is north of Dim Corner and east of Dome. Dim Corner is southeast of Dome and east of South Chapel. Ruined Corner is southwest of Dome, west of South Chapel, and south of Western End.

The church door is east of Eastern End and west of the Courtyard. The church door is a door.

Test me with "back / n / go back / e / open door / go through door / go back".

---

104 **Example** **No Relation** *RB*

A car which must be turned on before it can be driven, and can only go to roads.

We need to designate certain rooms as roads. Since the status of being a road will not change during play, we do this with a kind:

"No Relation"

A road is a kind of room. Definition: a room is offroad if it is not a road.

```
Instead of going by a vehicle (called the auto) to
somewhere offroad:
    say "You can't drive [the auto] off-road."


Trafalgar Square is a road. "The Square is overlooked
by a pillared statue of Admiral Lord Horatio Nelson (no
relation), naval hero and convenience to pigeons since
1812."


The National Gallery is north of Trafalgar Square. The
Strand is east of Trafalgar Square. The Strand is a road.


The car is a vehicle in Trafalgar Square. The ignition is
a device. The ignition is part of the car. Instead of going
by the car when the ignition is switched off: say "The
ignition is off at the moment." Instead of switching on
the car, try switching on the ignition. Instead of switching
off the car, try switching off the ignition.


Test me with "get in car / n / e / turn on car / n / e / get
out / w / n / s / e / get in car / turn off car / w / turn on
ignition / w".
```

(In the course of the writing of Inform 7, much of Trafalgar Square was pedestrianised, making this example already out of date.)

A further technical note: notice "going by a vehicle" in the above rule, rather than "going by something". A rule such as "Instead of going by something..." will be matched whenever the player tries to go some direction while in an enterable object, whether or not that object is actually capable of movement. This is sometimes useful, but in this case we want the warning to apply only when the player is in a vehicle; if we added Trafalgar Square's statue bases to the scenario, we would not want

```
You can't drive the pedestal off-road.
```

So we restrict the rule to "Instead of going by a vehicle..."

---

105 Example: Mattress King
Adding extra phrasing to the action to PUSH something in a direction. *RB*

By default, when the player pushes something a direction, Inform checks to make sure that the object is pushable between rooms. If not, it blocks the action; if so, it carries out a normal going action with the pushed object taken along.

Also by default, this action produces only a description of the new room that we've traveled into. But suppose we would like to print a short message describing the pushing action first:

```
"Mattress King"


Monica's Bedroom is a room. The Living Room is south
of Monica's Bedroom. Rachel's Bedroom is south of the
Living Room.


After going a direction (called way-pushed) with
something (called the thing-pushed):
    say "You push [the thing-pushed] [way-pushed] to
[the location].";
    continue the action.


The race car bed is an enterable supporter in Monica's
Bedroom. It is pushable between rooms.


Test me with "push bed south".
```

Example: One Short Plank

RB

A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

```
"One Short Plank"

The East Jungle is a room. The plank bridge is west of
the East Jungle and east of the West Jungle. The plank
is an open unopenable door. "A precarious plank bridge
extends [if the location is West
Jungle]east[otherwise]west[end if] across the chasm."
The description of the plank is "Extremely fragile and
precarious."

Instead of going through the plank when the player is
carrying something:
    say "You step gingerly across the plank, which bows
under your weight. But your meagre possessions are
the straw which breaks the camel's back!";
    end the story.

After going through the plank:
    say "You step gingerly across the plank, grateful that
you're not burdened.";
    continue the action.

There is a feather in the East Jungle.
```

But indeed, why stop there?

```
The gigantic stone ball is a thing in the West Jungle. It is
pushable between rooms.
```

```
Before going through the plank with something:
    say "Surely you jest." instead.


Test me with "w / e / w / push ball e / e / get feather / w".
```

 ExampleProvenance Unknown *RB*

Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

Suppose we have a series of items that might be stacked on top of one another -- say a heavy television on a rolling cart, and we want the player to be able to move the cart with PUSH TELEVISION EAST just as well as with PUSH CART EAST.

This takes a little redirection, using a setting action variables rule. This is not a kind of rule we've encountered yet, and in fact we won't meet it until the Advanced Actions chapter; it is included here for the convenience of authors who want to modify the effect of pushing without reading that far ahead:

```
"Provenance Unknown"


Setting action variables for pushing something to:
    if the noun is enclosed by a pushable between rooms
thing (called the pushed item) which is in the location:
        now the noun is the pushed item instead.
```

This rule says that any time we push an object that is on top of a stack of pushable objects, we should transfer the action to the item at the bottom of the stack.

The rest is merely a test case.

```
The heavy golden idol is on a roller board. The roller
board is on a hovercraft.
```

The hovercraft, the tea trolley, and the skateboard are pushable between rooms.

The hovercraft is in Zeta Proximan Dig Field.

Zeta Proximan Dig Field is a room. "During the day, the field is massed with sweating native workers, overseers, and officials from central command. Now the spades, trowels, brushes, metal detectors, ground probes, plumb lines, and sighting tripods have been laid aside.

All that remains are the trenches and the fine grey dust that blows slowly across them; the moonlight; and the just-emerging outlines of an ancient and alien wall."

The Hover-Road is west of the Dig Field. "A long road hastily laid down, stretching east to west, from the dig site toward the safety of the city."

When play begins:
    say "You have, at last, loaded your illicit cargo without setting off any of the many and sensitive alarms set here; now it remains only to sneak out of the area, under the light of Zeta Proxima's lone green moon."

Test me with "push idol west / look / push roller board east / look".

ExampleZorb
Replacing the message the player receives when attempting to push something that isn't

There are two aspects of Inform's handling of pushable objects that are particularly prime for modification. One is that we may want to change the language used to refuse the pushing of unpushable objects.

Second, Inform by default assumes that it is impossible to push objects in up or down directions. This makes lots of sense if the player is trying to push a wheelbarrow up a ladder; it makes less sense if instead we're pushing a ball up a slope.

We solve both problems with some syntax borrowed from the chapter on rulebooks: in the first case, we replace the old rule with a new one with more friendly phrasing; in the second, we remove the rule entirely. More about how to do this is described in the rulebooks chapter; and in general we can find out what rules contribute to any given action by looking at the Actions index. In this case, the action is "pushing it to", which has its own set of prerequisites (called check rules) that make sure the object can safely be pushed, before turning processing over to the going action.

```
"Zorb"

Section 1 - Procedure

The new can't push unpushable things rule is listed
instead of the can't push unpushable things rule in the
check pushing it to rules.

This is the new can't push unpushable things rule:
    if the noun is not pushable between rooms:
        say "[The noun] [are] not amenable to being
pushed from place to place." instead.

The can't push vertically rule is not listed in any
rulebook.
```

And now to provide a scenario where the player can push something up and down a hillside. Most of the rest of the example is there for local color and to provide a way to demonstrate these rule adjustments:

```
Section 2 - Scenario


The Steep Hill is a room. The Crest is above Steep Hill.
The Valley is below Steep Hill.


The flat rock is a fixed in place thing in the Steep Hill.


The Zorb is a transparent open enterable container in
the Steep Hill. "[if the player props the Zorb]The Zorb
rests here, kept from further rolling by your
support[otherwise]The Zorb is here[end if].". It is
pushable between rooms. The description of the Zorb is
"A giant plastic inflatable ball, like a hamster ball for
humans[if someone is in the Zorb]. Inside [is-are list of
people in the Zorb][end if]."


Lucy is a woman in the Zorb.


Carry out going with the Zorb when the Zorb contains
Lucy:
    say "Lucy whoops delightedly as she rides along in
the Zorb."


Every turn when the Zorb is not in the Valley and the
player does not prop the Zorb:
    let next room be the room down from the location of
the Zorb;
    if the player is not in the Zorb and the player can see
the Zorb:
        say "The Zorb succumbs to gravity and rolls down
```

```
toward [the next room].";
    move the Zorb to the next room;
    if the player is in the Zorb:
        say "The Zorb rolls you down the hill!";
        try looking;
    otherwise if the player can see the Zorb:
        say "The Zorb rolls ponderously but inevitably into
the vicinity.";


Propping relates one person to one thing. The verb to
prop means the propping relation.


Carry out going with the Zorb:
    now the player props the Zorb.


Before doing something when the action requires a
touchable noun:
    if the noun is not the Zorb, now the player does not
prop the Zorb.


Check waving hands when the player is propping
something (called casualty):
    try the player releasing the casualty.


Carry out entering the Zorb:
    now the player does not prop the Zorb.


Understand "let go of [something]" or "let [something]
go" or "release [something]" or "free [something]" as
releasing. Releasing is an action applying to one thing.


Check releasing:
    if the player carries the noun:
        try dropping the noun instead.
```

```
Check releasing:
    if the player does not prop the noun:
        say "You are not supporting [the noun]." instead.

Carry out releasing:
    now the player does not prop the noun.

Report releasing:
    say "You let go of [the noun]."

Test me with "d / push zorb up / look / push zorb up /
wave / d / d / push zorb up / release zorb / d / push zorb
up / touch rock / push the flat rock south".
```

---

109 Example Dearth and the Maiden *RB*

Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and seemliness.

The following example, indebted to the late Georgette Heyer, is suggestive:

```
"Dearth and the Maiden"

The Chequers Inn is a room. "The room is panelled and
ceilinged in oak, with blue curtains to the windows and
blue cushions on the high-backed settle by the fire."

An oil painting is in the Inn. "An oil painting hangs upon
one wall, a lascivious work from the Indies in which a
```

very bendy, sloe-eyed courtesan - but no."

A man called Mr Carr is in the Inn. "Standing bashfully aside is one Mr Carr, who we have been led to understand is by profession a Highwayman (yet whose visage oddly recalls Lord John Carstares, disgraced eldest son of the Earl of Wyncham)."

Kissing Mr Carr is unmaidenly behaviour. Doing something to the painting is unmaidenly behaviour.

Instead of unmaidenly behaviour in the Inn, say "How unmaidenly! Why, one might just as wantonly strip a rose of its petals, letting each fragrant leaf flutter slowly to the ground."

Test me with "examine painting / take painting / kiss mr carr".

---

110 **ExampleMimicry** *RB*

People who must be greeted before conversation can begin.

Suppose we want to add a sense of some conversational flow, so that the player is forced to acknowledge the presence of people before beginning detailed conversations with them. We collect all speech actions into a single category:

"Mimicry"

Asking someone about something is speech. Telling someone about something is speech. Answering

```
someone that something is speech. Asking someone for
something is speech.
```

And then write a general rule.

```
Before speech in the presence of an ungreeted person:
try waving hands.
```

One complication is that "asking someone to try doing something", which describes commands such as FRED, GO SOUTH, cannot be made into a kind of action. This requires its own rule:

```
Before asking someone to try doing something in the
presence of an ungreeted person: try waving hands.
```

Now we define what greetings are going to look like:

```
Check waving hands:
    unless the player can see someone who is not the
player, say "You are alone." instead.
```

```
Carry out waving hands:
    say "You nod hello to [the list of ungreeted people
who can be seen by the player].";
    now every ungreeted person who can be seen by the
player is greeted.
```

```
The report waving hands rule is not listed in the report
waving hands rulebook.
```

Because of the way we've defined the command, this will now also work if the player waves.

```
A person can be greeted or ungreeted. A person is
usually ungreeted. The player is greeted.
```

And now the scenario:

```
The International Convention of Mimes is a room.
Lester, Harold, Geoff, Kwame, and Peter are men in the
Convention. Elouise is a woman in the Convention. The
Invisible Box is an enterable container in the
Convention. "You can detect, from the way people keep
leaning on it, an invisible box in the middle of the room."



Lester carries a bowler hat.



Instead of speech in the presence of someone:
    describe poor reception.



Definition: a person is other if it is not the player.



At 9:01 AM:
    move Phineas to the location; say "A mime called
Phineas appears from the non-existent bathroom."



Phineas is a man.



A persuasion rule:
    describe poor reception;
    persuasion fails.



To describe poor reception:
    if the player is in the Invisible box,
        say "Everyone convulses with silent laughter as
you try to shout from within the invisible box.";
    otherwise
        say "You attempt to convey your meaning with
```

gesture and interpretive dance, but [the list of visible other people] scorn[if the number of visible other people is 1]s[end if] your performance, refusing to respond."

Test me with "ask lester about work / lester, east / ask lester for bowler / lester, nice not talking to you / get in box / ask lester for hat / phineas, east".

---

111 **Example Y ask Y?** *RB*

Noticing when the player seems to be at a loss, and recommending the use of hints.

Suppose we'd like to watch for signs that the player is floundering, and if we see them, recommend that he try the hints. There are probably more sophisticated diagnostics, but as a first cut, let's assume that a player who repeatedly reviews descriptions of objects he's already seen, looks around the room, and takes inventory, is at a loss for more productive activities. So then...

"Y ask Y?"

A thing can be examined or unexamined. A thing is usually unexamined. Carry out examining something: now the noun is examined.

Taking inventory is acting confused. Looking is acting confused. Examining an examined thing is acting confused.

After acting confused for the sixth turn:
    say "(If you are feeling lost, try typing HELP for suggestions.)"

And now we write a scenario which will, alas, rather encourage even a

deft and clueful player to play as though he were hopelessly confused:

```
The story headline is "or: Bad Author, No Biscuit".


The description of a thing is usually "Hm. [The item
described] reminds you quite a lot of [a random visible
thing which is not the item described]."


The Yurt is a room.


Food is a kind of thing. Food is always edible. In the
Yurt are a yam and a dish of yakitori. The yam and the
yakitori are food. The description of food is "Well, at
least it's not [a random edible thing which is not the item
described]."


In the Yurt is an animal called a yapok.


The player wears a yukata. The player carries a
yataghan.


Every turn:
    if a random chance of 1 in 2 succeeds and something
is examined:
        say "Your eye is attracted by some kind of
surreptitious movement from [the random examined
thing].";
    otherwise if the player carries something and a
random chance of 1 in 3 succeeds:
        say "[The random thing carried by the player] tries
to slip from your grasp."
```

```
Test me with "x yam / x yam / look / x yam / i / look / i /
help / quit".
```

And finally a little dollop of perversity from a later chapter:

```
Check quitting the game:
    say "You're sure? ";
    if player consents, say "[line break]You were getting
close to a breakthrough, you know.[line break]";
    otherwise stop the action.


Understand "help" as a mistake ("You're doing fine! Just
keep at what you're doing now.").
```

---

112 **Example** **A Day For Fresh Sushi** *RB*

A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is performed.

The following is an almost-completely-faithful rewrite of Emily Short's "A Day for Fresh Sushi", which was originally written using the (very different) Inform 6 programming language. The -- let us be honest and call it a gimmick -- of this game is the evil fish, who has some unpleasant remark to offer on pretty much every action. But the effect would wear off fast if he repeated himself, so these comments need to be single-use only.

Inform 7's repeated action syntax makes it much tidier to write the same scenario, so:

```
"A Day For Fresh Sushi" by Emily Short.


Use scoring.
```

The story headline is "Your basic surreal gay fish romance".

The Studio is a room. "[if visited]Decorated with Britney's signature flair. It was her innate sense of style that first made you forgive her that ludicrous name. And here it is displayed to the fullest: deep-hued drapes on the walls, the windows flung open with their stunning view of old Vienna, the faint smell of coffee that clings to everything. Her easel stands over by the windows, where the light is brightest.[otherwise]This is Britney's studio. You haven't been around here for a while, because of how busy you've been with work, and she's made a few changes -- the aquarium in the corner, for instance. But it still brings back a certain emotional sweetness from the days when you had just met for the first time... when you used to spend hours on the sofa...

You shake your head. No time for fantasy. Must feed fish.[end if]"

Instead of smelling the Studio:
    say "The evil fish notices you sniffing the air. 'Vanilla Raspberry Roast,' it remarks. 'You really miss her, don't you.'

You glance over, startled, but the fish's mouth is open in a piscine equivalent of a laugh. You stifle the urge to skewer the thing..."

Instead of jumping:
    say "'Er,' says the fish. 'Does that, like, EVER help??'"

Instead of going nowhere:
    say "You can't leave until you've fed the fish.

Otherwise, he'll complain, and you will never hear the end of it."

The cabinet is an openable closed container in the Studio. It is fixed in place. "A huge cabinet, in the guise of an armoire, stands between the windows." The description is "Large, and with a bit of an Art Nouveau theme going on in the shape of the doors." Understand "armoire" as the cabinet.

Instead of looking under the cabinet for the first time:
    say "'Dustbunnies,' predicts the fish, with telling accuracy. It executes what for all the world looks like a fishy shudder. 'Lemme tell you, one time I accidentally flopped outta the tank, and I was TWO HOURS on the floor with those things STARING ME IN THE NOSE. It was frightening.'"

After opening the cabinet for the first time:
    say "'There ya go,' says the fish. 'The girl is getting WARMER.'"

After closing the cabinet for the first time:
    if the fish food is not found, say "'Ooh, what do you think, Bob? I think we're going to have to dock the girl a few points. HAVE ANOTHER LOOK, sweetcakes, there's a doll.'"

The cabinet contains some paints and some cloths. The description of the paints is "A bunch of tubes of oil paint, most of them in some state of grunginess, some with the tops twisted partway off."

After taking the paints for the first time:
    say "'Boy,' says the fish, apparently to himself, 'I sure hope that's some food she's finding for me in there. You

know, the yummy food in the ORANGE CAN.'"

After examining the paints for the first time:
    say "'Tons of useful stuff in there,' hollers in the fish, in a syncopated burble."

The description of the cloths is "Various colors of drapery that Britney uses to set up backgrounds and clothe her models. She does a lot of portraiture, so this comes in handy. It's all a big messy wad at the moment. Organized is not her middle name." Understand "drapery" or "cloth" as the cloths. The indefinite article of the cloths is "a heap of". [see 3.17]

Instead of searching or looking under the cloths for the first time:
    now the player is carrying the fish food;
    now the fish food is found;
    say "Poking around the cloths reveals -- ha HA! -- a vehemently orange can of fish food."

Instead of showing the cloths to the fish:
    say "'What are you, some kind of sadist? I don't want to see a bunch of cloths! What kind of f'ing good, 'scuse my French, is that supposed to do me? I don't even wear pants for God's sake!'

He really looks upset. You start wondering whether apoplexy is an ailment common to fish."

After examining cloths for the first time:
    say "'Whatcha looking at? I can't see through the doors, you know.'"

There is a can of fish food. Understand "canister" as the can. The description is "A vehemently orange canister of fish food." The fish food can be found or hidden. The fish food is hidden.

Instead of giving the can to the fish:
    say "'I don't want the whole can, GeniusChyk. Just feed me and we'll ALL be happy, 'kay?"

Instead of showing the can to the fish:
    say "'That's the ticket, sweetie! Bring it on.'"

Instead of opening the can:
    say "'Oh, for--!' The evil fish breaks out in exasperation and hives. 'Screw the screwing around with the screwtop. SHE never has to do that.'

'Well, SHE is not here,' you reply. 'What do you suggest?'

'>FEED FISH<' says the fish promptly, making fishy faces and pointing at you with his fin. 'Simplicity. Try it.'"

Instead of inserting the can into something:
    say "'HelLLLOOO,' screams the fish. 'Whatever happened to FEEDING MEEE?'"

The easel is a supporter in the Studio. It is scenery. On the easel is a painting. Understand "portrait" or "image" as the painting.

The description of the painting is "Only partway finished,

but you can tell what it is: Britney's mother. You only met the old woman once, before she faded out of existence in a little hospice in Salzburg.

In the picture, her hands are grasping tightly at a small grey bottle, the pills to which she became addicted in her old age, and strange, gargoyle-like forms clutch at her arms and whisper in her ears.

But the disturbing thing, the truly awful thing, is the small figure of Britney herself, down in the corner, unmistakable: she is walking away. Her back turned.

You thought she'd finally talked this out, but evidently not. Still feels guilty for leaving. You only barely stop yourself from tracing, with your finger, those tiny slumped shoulders..."

Instead of taking the painting, say "No, you'd better leave it. It'd freak her out if you moved it."

Before examining the painting for the first time:
    say "A ferocious banging from the aquarium attracts your attention as you go to look at the painting. 'Hey!' screams the fish. 'She doesn't like strangers looking at her paintings before they're DOONNNE!'

'Shut up, you,' you reply casually. 'I'm not a stranger.' But the fish puts you off a little bit, and your heart is already in your mouth before you see the painting itself...".

Instead of examining the painting more than once:
    say "Once is really enough. It's pretty much

embedded in your consciousness now."

After doing something to the painting:
    say "'So what's it of?' asks the fish, as you turn away. 'She never asks if I want to see them, you know?'

'Her mother,' you respond without thinking.

'Yeah? Man. I never knew my mother. Eggs, that's the way to go.'"

The window is scenery in the Studio. The window can be openable. The window can be open. It is openable and closed. Understand "windows" as the window. The description of the window is "[if open]Through the windows you get a lovely view of the street outside. At the moment, the glass is thrown open, and a light breeze is blowing through.[otherwise]Through the windows, you get a lovely view of the street outside -- the little fountain on the corner, the slightly dilapidated but nonetheless magnificent Jugendstil architecture of the facing building. The glass itself is shut, however.[end if]"

After opening the window for the first time:
    say "'Thank god some air,' says the fish. 'Man, it was getting hard to breathe in here.' Two beats pass. 'Oh wait.'"

The table is scenery in the Studio. On the table is a vase. The vase is an open container. It is not openable.

The description of the table is "A monstrosity of poor taste and bad design: made of some heavy, French-

empire sort of wood, with a single pillar for a central leg, carved in the image of Poseidon surrounded by nymphs. It's all scaley, and whenever you sit down, the trident has a tendency to stab you in the knee. But Britney assures you it's worth a fortune." The description of the vase is "A huge vase -- what you saw once described in a Regency romance as an epergne, maybe -- something so big that it would block someone sitting at the table from seeing anyone else also sitting at the table. But it does function nicely as a receptacle for hugeass bouquets of flowers."

Instead of looking under the table for the first time:
    say "'You're not going to find anything down there,' whines the fish. 'I mean, c'mon. It's the fricking floor. Please tell me you can see that. I can see that. I'm a myopic fish in a tank ten feet away and I can tell you there is nothing there but floor.'"

After examining the table:
    say "'That there is MY PA,' says the fish, pointing at the scaley triton figure with one fin."

Instead of inserting something which is not the bouquet into the vase:
say "'Okay, so, what were you, raised in a barn? Normal folks like to use that for flowers. Or so I've observed.'"

After inserting the bouquet into the vase for the first time:
    say "You settle the flowers into the vase and arrange them so that they look sprightly.

'Oooh,' says the fish. 'No one ever changes the plant life in HERE. It's the same seaw--'

'Cut me a break and cork it,' you reply tartly."

The player is carrying a telegram, a bouquet, and a lingerie bag. The player is wearing a chef hat.

The description of the telegram is "A telegram, apparently. And dated three days ago. [fixed letter spacing]TRIUMPH OURS STOP BACK SOON STOP BE SURE TO FEED FISH STOP[variable letter spacing]". [For printing options see 4.13.] Understand "yellow paper" as the telegram.

After examining the telegram for the first time:
    say "'So,' blubs the evil fish. 'How about it? Little food over here?'"

After examining the telegram:
    choose a random row in the Table of Insulting Fish Comments;
    say "[comment entry][paragraph break]".

Table of Insulting Fish Comments
comment
"'Yeah, yeah,' says the fish. 'You having some trouble with the message, there? Confused? Something I could clear up for you?'"
"'Ookay, genius kid has some troubles in the reading comprehension department.' The fish taps his head meaningfully against the side of the tank. 'I'm so hungry I could eat my way out, you get my meaning?'"
"'I'll translate for you,' screams the fish in toothy fury. 'It says GIVE FOOD TO FISH!! How much more HELP do you NEED???"

The description of the chef hat is "A big white chef hat of the kind worn by chefs. In this case, you. Just goes to

show what a hurry you were in on the way out of the restaurant." Understand "big" or "white" or "chefs" or "chef's" as the chef hat. [Inform knows that this is clothing because the player starts out wearing it, so there's no need to say so separately.]

The aquarium is a transparent open container in the Studio. It is not openable. "In one corner of the room, a large aquarium bubbles in menacing fashion." The description of the aquarium is "A very roomy aquarium, large enough to hold quite a variety of colorful sealife -- if any yet survived." Understand "tank" as the aquarium.

The aquarium contains some gravel and some seaweed. Understand "little rocks" as the gravel. Understand "weed" as the seaweed. The description of the gravel is "A lot of very small grey rocks." The description of the seaweed is "Fake plastic seaweed of the kind generally bought in stores for exactly this purpose."

The examine containers rule does nothing when examining the aquarium.

After examining the gravel for the first time:
    say "The fish notices your gaze; makes a pathetic mime of trying to find little flakes of remaining food amongst the gravel."

After examining the seaweed for the first time:
    say "'Nice, hunh?' blubs the fish, taking a stabbing bite out of one just by way of demonstration. 'Look so good I could eat it.'"

The aquarium contains an animal called an evil fish. The description of the fish is "Even if you had had no prior

experience with him, you would be able to see at a glance that this is an evil fish. From his sharkish nose to his razor fins, every inch of his compact body exudes hatred and danger."

Instead of taking the evil fish:
    say "The fish swims adroitly out of range of your bare hand. 'Hey,' he says, and the bubbles of his breath brush against your fingers. 'Count yourself lucky I don't bite you right now, you stinking mammal.'"

Instead of attacking the evil fish:
    say "Oh, it's tempting. But it would get you in a world of hurt later on."

Instead of kissing the evil fish:
    say "You're saving all your lovin for someone a lot cuddlier."

After examining the evil fish for the first time:
    say "The fish glares at you, as though to underline this point."

After examining the evil fish for the second time:
    say "'If you're looking for signs of malnutrition,' says the fish, 'LOOK NO FURTHER!!' And it sucks in its gills until you can see its ribcage."

An every turn rule:
    choose a random row in the Table of Fish Banter;
    say "[comment entry][paragraph break]".

Table of Fish Banter

used
ment
"Hey, nice SKIN TONE,' shouts the evil fish. His words reach you in a spitting gurgle of aquarium water. 'You gone over to a pure eggplant diet these days?'"
"The evil fish is floating belly up! ...oh, curse. He was toying with you. As soon as he sees you looking, he goes back to swimming around."
"The evil fish darts to the bottom of the tank and moves the gravel around with his nose."
"The evil fish is swimming around the tank in lazy circles."
"The evil fish begins to butt his pointy nose against the glass walls of the tank."

The description of the bouquet is "Okay, so it's silly and sentimental and no doubt a waste of money, of which there is never really enough, but: you miss her. You've missed her since ten seconds after she stepped aboard the shuttle to Luna Prime, and when you saw these -- her favorites, pure golden tulips like springtime -- you had to have them." Understand "flowers" or "tulip" or "tulips" as the bouquet.

After examining the bouquet for the first time:
    say "'Oh, you shouldn't have,' says the fish. 'For me??'
    You just respond with a livid glare."

    Instead of smelling the bouquet for the first time:
        say "'Mmm-mm,' says the fish. 'Damn, I sure wish I had olfactory abilities. Hey, if I did, I might be even better at noticing the presence or absence of FOOD.'"

    The description of the lingerie bag is "You grant yourself the satisfaction of a little peek inside. You went with a pale, silky ivory this time -- it has that kind of sophisticated innocence, and it goes well with the purple of your skin. A small smirk of anticipation crosses your lips."

After examining the lingerie bag for the first time:
say "'What's in THERE?' asks the fish. 'Didja bring me take-out? I don't mind Chinese. They eat a lot of carp, but what do I care? I'm not a carp. Live and let live is what I s--'

'It's NOT take-out.' You stare the fish down and for once he actually backstrokes a stroke or two. 'It's PRIVATE.'"

After examining the lingerie bag for the second time:
say "'If it's not take-out, I don't see the relevance!' shouts the fish. 'Food is what you want in this situation. Food for MEEEE.'"

Understand the command "feed" as something new.

Understand "feed [something]" as feeding.

Feeding is an action applying to one visible thing.

Check feeding:
if the noun is not the evil fish, say "That doesn't make much sense." instead;
if the player is not carrying the fish food, say "You need the fish food first." instead.

Carry out feeding:
increment the score;
say "Triumphantly, you dump the remaining contents of the canister of fish food into the tank. It floats on the surface like scum, but the fish for once stops jawing and starts eating. Like a normal fish. Blub, blub.[paragraph break]";
say "[bold type] *** TWO HOURS LATER ***[roman

type][paragraph break]'So,' Britney says, tucking a strand of hair behind your ear, 'where shall we go for dinner? Since I made the big bucks on this trip, it's my treat. Anywhere you like.'[paragraph break]'I've had a hankering all day,' you admit, as the two of you turn from the shuttle platform and head toward the bank of taxis. 'I could really go for some sashimi right now.'";
    end the story finally.

Before feeding the fish food:
    try feeding the evil fish instead.

When play begins:
    say "You're on the run. You've got a million errands to do -- your apartment to get cleaned up, the fish to feed, lingerie to buy, Britney's shuttle to meet-- [paragraph break]The fish. You almost forgot. And it's in the studio, halfway across town from anywhere else you have to do. Oh well, you'll just zip over, take care of it, and hop back on the El. This'll be over in no time.[paragraph break]Don't you just hate days where you wake up the wrong color?[paragraph break]".

The maximum score is 1.

Test me with "x fish / g / kiss fish / x aquarium / x gravel / x seaweed / i / x telegram / x bouquet / smell bouquet / x lingerie / g / x hat / x window / open window / x painting / g / x cabinet / open cabinet / x cloths / search cloths / open food / feed fish".

Chapter 8: Change

s; *§8.9. Moving the player*; *§8.10. Removing things from play*; *§8.11. Now...*; *§8.12. Increasing and dec*

Contents of *Writing with Inform*
Chapter 7: Basic Actions
Chapter 9: Time
Indexes of the examples

§8.1. Change of values that vary

So far, what we have done in response to the player's commands amounts to little more than a few ripostes. The simulated world does change during play, as the player moves from room to room or picks up things, but all of this is happening automatically, not at our direct instruction. How then can we make the world change?

Recall that the world consists of rooms, in which are things, and that all of these have properties appropriate to their kinds. Some properties are either/or ("open" or "closed" but not both and not neither), while others have values (the "matching key" of a lockable door, for instance). Finally, we may also have created some free-standing values or "variables".

We take the last example first, as it is the simplest. Suppose we have:

```
"Winds of Change"

The prevailing wind is a direction that varies. The prevailing wind is
northwest.

The Blasted Heath is a room. "Merely an arena for the play of witches
and kings, my dear, where the [prevailing wind] wind blows."

Instead of waiting when the prevailing wind is northwest:
```

```
    say "A fresh gust of wind bowls you over.";
    now the prevailing wind is east.
```

The new phrase here is "now". This automatically checks that the new value is one which makes sense in the given context, so for instance it would not allow either of these:

```
now the prevailing wind is 25;
now the prevailing wind is the Heath;
```

the former being a number, and the latter a room, so that neither is a direction. Similarly, "now" will not allow constant values to be changed. So

```
Colour is a kind of value. The colours are blue, red and mauve.

After pulling the psychedelic lever:
    now blue is mauve.
```

...will result in a problem message; it's like writing "now 1 is 2". The difference between "the prevailing wind" and "blue" is that the wind was declared to be a "direction that varies", whereas blue wasn't.

Start of Chapter 8: Change

Back to Chapter 7: Basic Actions: §7.18. Postscript on actions

Onward to §8.2. Changing the command prompt

§8.2. Changing the command prompt

The command prompt is the text printed by Inform to ask the player for another command. Ordinarily this is simply a greater-than-sign, ">", so we tend not to notice it as text at all. Internally, though, it is a variable value called "command prompt", which means we can change it.

For example, this will be a more conversational sort of prompt:

When play begins: now the command prompt is "What now? ".

Whereas this will be more up-to-the-minute and demanding:

When play begins: now the command prompt is "[time of day] >".

("Time of day" is another variable value, which is fairly self-explanatory, but will be covered in detail later on.) The prompt can be changed at any point, so can be used to indicate the current situation, or even as a sly way to introduce a sort of conversation between computer and player.

Start of Chapter 8: Change

Back to §8.1. Change of values that vary

Onward to §8.3. Changing the status line

Example 113: ★★★ **Don Pedro's Revenge** Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

§8.3. Changing the status line

The status line is the black bar along the top of a story being played, which ordinarily displays the current position; in a story with scoring, it also usually shows the score and number of moves taken. Like the command prompt, it is not fixed but results from values which can be changed: the "left hand status line" and "right hand status line".

The default values are "[the player's surroundings]" for the left hand status line and "[score]/[turn count]" for the right hand status line (if there's scoring; it's blank otherwise). Score and turn count are numbers which vary in play (more about scoring later); "[the player's surroundings]" is a text substitution really intended for just this purpose:

```
say "[the/-- player's surroundings]"
```

This text substitution produces a succinct description of where the player is, be this in darkness, in a lighted room or inside an opaque container such as a large packing case. Example:

```
now the left hand status line is "You: [the player's
surroundings]";
```

These make useful elements to juggle in redesigning the status line, as in the following example:

```
When play begins:
    now the left hand status line is
        "[the player's surroundings] / [turn count] / [score]";
    now the right hand status line is "Time: [time of day]".
```

The text in the right hand status line should be kept no more than 14 letters long, including any spaces. The left hand status line has more leeway, but should still be kept brief.

See **Awarding points** *for scoring*

Start of Chapter 8: Change

Back to §8.2. Changing the command prompt

Onward to §8.4. Change of either/or properties

Example 114: **Politics as Usual** Have the status line indicate the current region of the map.

Example 115: **Centered** Replacing the two-part status line with one that centers only the room name at the top of the screen.

---

§8.4. Change of either/or properties

When we have an either/or property, we can set it like so:

```
Instead of waiting when the oaken door is closed:
    say "There is a slow, creaky click! sort of noise as the door swings
open, apparently all by itself.";
    now the oaken door is open.
```

If it is open already, nothing changes: in any case nothing is said to the player unless we give explicit instructions to that effect, as we've done here.

Inform protects its model world from accidental damage in several ways, one of which is to ensure that things are not given properties which they are not allowed to have. So this, for instance, will not be accepted:

```
now the oaken door is unvisited
```

More subtle problems arise if it is not possible to tell, when the story is being constructed, what the object in question will be: for instance, if we try to change a randomly chosen object to be "unvisited". Inform therefore makes additional checks during play, printing up a suitable message only if the rules are violated. The net effect is that it is impossible for the oaken door ever to have the "unvisited" property.

Start of Chapter 8: Change

Back to §8.3. Changing the status line

Onward to §8.5. Change of properties with values

Example 116: **Vitrine** An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

---

§8.5. Change of properties with values

Changing properties with values is very similar:

```
now the printed name of the Closet is "Suddenly Spooky Closet"
```

Inform checks three different things to ensure that this change is safe to perform. Firstly, the value must be the right kind for the property in question, so this for instance would be rejected:

```
now the printed name of the Closet is 7
```

Secondly, the object in question has to be allowed to have the given property. This, for instance, would be disallowed:

```
now the initial appearance of the Closet is "Dusty"
```

(since "initial appearance" is a property which only things can have, not rooms). Finally, the object has to actually have the property, not just have the right to have that property. Thus:

```
now the printed name of the Closet is "Suddenly Spooky Closet"
```

...is only permitted if the Closet is designed with a "printed name". In fact this is certain to be true: all rooms and things automatically have a printed name, which is the short boldface description in the case of rooms, and the usual text briefly describing something in the case of things.

"Now" is a simple way to change many things in Inform, but it's cumbersome to change the map of the model world using "now", because the map is such a complicated arrangement. (It's not a property: it's a sort of mesh of relations.) So a special phrase exists to change map connections:

change (direction) **exit of** (room) **to** (room)

This phrase alters the map so that the given map connection is made. Note that connections can be made to rooms, but not doors: the positions of doors are fixed. Example:

```
change the east exit of the Closet to the Tsar's Imperial
Dining Salon
```

Since "nothing" is not a room, this doesn't allow us to change the exit to

nothing, so there is a separate definition of:

```
change the west exit of the Closet to nothing
```

change (direction) **exit of** (room) **to nothing**/nowhere

This phrase alters the map so that the given map connection is unmade. Example:

```
change the west exit of the Closet to nowhere
```

Altering the map itself is not a very subtle way to adjust when and where the player can move - writing suitable rules is usually a cleaner solution - so this phrase is best avoided unless really needed.

Start of Chapter 8: Change

Back to §8.4. Change of either/or properties

Onward to §8.6. Whose property?

Example 117: **Thirst** A waterskin that is depleted as the player drinks from it.

Example 118: **Thirst 2** A campfire added to the camp site, which can be lit using tinder.

---

§8.6. Whose property?

This seems a useful point to clarify something already seen. We normally call a property with a value something like:

```
the printed name of the West Ballroom
```

We are sometimes allowed to omit the "of the ..." part, and simply call it "the printed name", for the sake of brevity. For instance, the following room description:

> The West Ballroom is a room. "A handsome sweep of chequered floor beckons the eye into the [printed name]."

will result in "West Ballroom" being substituted for "[printed name]". Since the text belongs to the West Ballroom, that is assumed to be the owner of any properties named in its description. Similarly:

> Instead of examining something, say "Hmm, let me see: [printed name]..."

Here the owner of the "printed name" is assumed to be the noun referred to in the action - in other words, the "something" alluded to in the rule.

Start of Chapter 8: Change
Back to §8.5. Change of properties with values
Onward to §8.7. Moving things

§8.7. Moving things

We have now seen how to change the properties of rooms and things, and also any freestanding values which may have a bearing on the model world. We are not allowed to change the kind of anything during play. Our remaining freedom is to move things around. It would make no sense to move rooms around, because rooms are the fixed reference points in our geography, but anything else is mobile. This even includes things which are supposedly "fixed in place", for unlike the player, we have god-like powers. (There are minor restrictions: backdrops are trickier to move, since they are present in several rooms at once - see the next section. And doors, at the junction between two rooms, cannot be moved.)

Here is how to move something:

move (object) **to** (object)

This phrase moves the first-named object to the second. Example:

```
move the genie's lamp to Aladdin's Cave;
```

The first object named has to be a thing; the destination must be a room, as here, a container, a supporter, or a person. When something is moved, all its parts and contents (and all their contents, and so on) move with it. If the thing being moved is a person, then the destination is required to be a room or an enterable container. (In particular, a person cannot be carried by another person.)

Two options can be used if the object being moved is the player.

```
move the player to Aladdin's Cave, without printing a room
description
```

omits the description which would otherwise be produced. A compromise is to use:

```
move the player to Aladdin's Cave, printing an abbreviated
room description
```

which gives a full description if the player has never been here before, but only a brief one if it is a familiar scene. These options have no effect for any other objects being moved.

If the destination is a person, like so:

```
move the genie's turban to Aladdin;
```

then it will be carried rather than worn. We could arrange for it to be worn instead by writing

```
now the genie's turban is worn by Aladdin;
```

"Now..." is a much more flexible phrase than "move": more on this shortly.

Back to §8.6. Whose property?

Onward to §8.8. Moving backdrops

Example 119: ★★ **Meteoric I and II** A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

---

§8.8. Moving backdrops

A backdrop can be in several rooms at once. When created, its position can be given as any specific collection of rooms, or as a region, or even as "everywhere". For instance:

```
The Upper Cave is above the Rock Pool. The Ledge is east of the
Pool.

The stream is a backdrop. It is in the Upper Cave and the Ledge.
```

Moving backdrops is not like moving other things, because there's no single destination. There are several possibilities:

(a) A backdrop can be moved to a region. If we define:

```
Lower Level is a region. The Rock Pool and the Ledge are in the
Lower Level.
```

then we can write either of

```
move the stream to the Lower Level;
now the stream is in the Lower Level;
```

and either way, the stream is now found in the Rock Pool and the Ledge but nowhere else.

(b) A backdrop can be moved to a category of rooms:

move (object) **backdrop to all** (description of objects)

This phrase moves the backdrop so that it is now present in every room matching the given description. Example: If we define

```
A room can be wet or dry. A room is usually dry. The Rock
Pool is wet.
```

then we can write

```
move the stream backdrop to all wet rooms;
```

This phrasing, "move the ... backdrop to all ..." is deliberately meant to look unlike the simpler "move ... to ...", to emphasise that this kind of movement is possible only for backdrops.

What then happens is that the stream is present in whichever rooms are currently wet. But the stream's position is ordinarily checked only after movements, for efficiency's sake. So if the player is in a room which suddenly changes from being dry to being wet, the stream will not magically appear (though it will be there if the player goes out and comes in again). If this is not good enough, the phrase "update backdrop positions" can be used to ensure the accuracy of all backdrop locations after a dramatic change:

update backdrop positions

This phrase runs through all backdrops in the model world and makes sure they are correctly in, or not in, the current location, so that everything appears right from the player's point of view. Example:

```
The Upper Cave is above the Rock Pool. The Ledge is east
of the Pool. The stream is a backdrop.

When play begins:
```

```
    move the stream backdrop to all wet rooms.

A lever is in the Cave. The lever is fixed in place.

Instead of pulling the lever when the Cave is dry:
    now the Cave is wet;
    now the lever is in the Rock Pool;
    now the lever is portable;
    update backdrop positions;
    say "The old rusty lever pulls away, and the thin cave wall
goes with it, so that a stream bursts into the cave, falling to
the pool below."
```

(c) A backdrop can be moved to be either everywhere or nowhere:

```
After sleeping:
    say "It's a bright new day!";
    now the stars are nowhere.

After waiting:
    say "Darkness falls rapidly here.";
    now the stars are everywhere.
```

Start of Chapter 8: Change

Back to §8.7. Moving things

Onward to §8.9. Moving the player

Example 120: **Orange Cones** Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

§8.9. Moving the player

The player is a thing, too, and can also be moved, which has the effect of instantaneous transportation, without the need for a suitable map connection to the new location. For instance, these are equivalent:

```
move the player to the Bodleian Library;
now the player is in the Bodleian Library;
```

This will ordinarily result in a room description of the Bodleian Library being printed up, but that might not always be desirable. For instance:

```
Instead of waiting in the Schola Maleficorum:
    say "A bored demon catches your eye (they really do have very
inquisitive fingers) and throws you back out into the Antechamber.";
    move the player to the Antechamber, without printing a room
description.
```

Thus tacking on the option "without printing a room description", remembering to add the comma, omits the description which would otherwise be produced. A compromise is to use the option "printing an abbreviated room description": this gives a full description if the player has never been here before, but only a brief one if it is a familiar scene.

The player's point of view can also be moved by shifting to another character. Suppose the story features two people, Alice and Bob, and the player at the keyboard is giving commands to Alice, and seeing everything from her point of view. The phrase:

```
now the player is Bob
```

switches the perspective so that now Bob is the one controlled by the human player, and it's Bob's point of view which counts. The human being at the keyboard may feel a sense of having jumped abruptly from place to place, but in fact neither Alice nor Bob has moved.

A change of player can sometimes cause confusing things to happen, if it takes place as part of a successful action. Suppose there's an action called "possessing", which enables the player to possess somebody else's body; and suppose the player types POSSESS ADELE. The action succeeds, so that the player moves into the mind of Adele. But that means that at the end of the action, the player is no longer the actor - that is, no longer the person who began the action; and consequently, Inform won't

use the report rulebook to say what has just happened. It's a strange business, moving into another body.

Start of Chapter 8: Change

Back to §8.8. Moving backdrops

Onward to §8.10. Removing things from play

Example 121: ★★★ **Terror of the Sierra Madre** Multiple player characters who take turns controlling the action.

§8.10. Removing things from play

Some things will occasionally be in a limbo state called being "off-stage": like actors or props not needed in Act II, but perhaps to be brought back on-stage later, they wait on the sidelines. Anything created with no apparent location will start the story off-stage, as in the case of the lamp here:

```
Aladdin's Cave is a room. The genie's lamp is a container.
```

(Such things are easy to see in the World index because they are listed after all of the rooms and their contents, not belonging inside any room.) If we wanted to make this clearer to a human reader, we could add:

```
The lamp is nowhere.
```

to emphasise the point. In this context, "nowhere" means "in no room". Moving the lamp onto the stage-set, so to speak, is easy:

```
now the lamp is in the Cave;
```

or perhaps:

```
now the player is carrying the lamp;
```

and we can whisk it away again like so:

```
now the lamp is nowhere;
```

(We can't say "now the lamp is somewhere" because that's too vague about exactly where it is.) In older builds of Inform, the usual thing was to write "remove the lamp from play", but that's now a deprecated phrase: better to use "nowhere" instead.

remove (object) **from play**

Removes the given object from play, so that it is not present in any room. We are not permitted to remove rooms, or doors, or the player, from play; but we are permitted to remove backdrops, making them disappear from all rooms in which they are present. Example:

```
remove the gold coin from play;
```

We can test whether something is on-stage or off-stage with:

```
if the gold coin is somewhere, ...
if the gold coin is nowhere, ...
```

Inform also understands two adjectives for this:

```
if the gold coin is on-stage, ...
if the gold coin is off-stage, ...
```

Because these are adjectives, they can be used in a few ways which "nowhere" and "somewhere" can't, such as:

```
say "Ah, so many absent friends. Who now remembers [list of off-
stage people]?"
```

Note that "on-stage" and "off-stage" apply only to things. Rooms, directions and regions are the stage itself: so it makes no sense to ask the question of whether they are "on-" or "off-". Doors are always on-stage; a backdrop, say "the sky", is always

on-stage unless it has been taken off by writing something like "now the sky is nowhere".

Start of Chapter 8: Change

Back to §8.9. Moving the player

Onward to §8.11. Now...

Example 122: ★ **Beverage Service** A potion that the player can drink.

Example 123: ★ **Spring Cleaning** A character who sulks over objects that the player has broken (and which are now off-stage).

Example 124: ★★ **Extra Supplies** A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

---

§8.11. Now...

"Now" has already appeared several times in this chapter, being used like a Swiss army knife to change values of all kinds:

```
now the score is 100;
```

In fact, "now" is by far the most flexible phrase known to Inform.

now (a condition)

This phrase makes the condition become true. Examples:

```
now the score is 100;
now the player is Kevin;
now the front door is open;
now Mr Darcy is wearing the top hat;
now all the doors are open;
now all of the things in the sack are in the box;
```

Inform issues a problem message if the condition asks to do the impossible ("now 3 is an even number") or is vague ("now the duck is not in the Lily Pond") or not in the present tense ("now the front door had been open").

We've now seen all three things which can be done with a condition S which describes the world:

```
S. - The relation holds at the start of play.
if S, ...; - Does the relation hold right now?
now S; - Make the relation hold from now on.
```

For instance,

```
The apple is in the basket.
if the apple is in the basket, ...;
now the apple is in the basket;
```

Start of Chapter 8: Change

Back to §8.10. Removing things from play

Onward to §8.12. Increasing and decreasing

Example 125: ★ **Bee Chambers** A maze with directions between rooms randomized at the start of play.

Example 126: ★★ **Hatless** It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

Example 127: **Technological Terror** A ray gun which destroys objects, leaving their component parts behind.

---

§8.12. Increasing and decreasing

Once we begin to deal with named values (or table entries, list entries or other ways to describe places where values are kept), we find that we often want to change them. We could if we wanted always use "now" for this, but it can be a little clumsily worded if we want to increase or decrease something:

```
now the score is the score plus six;
```

Because of that, we have some convenient abbreviations which have the advantage that the value being changed only has to be named once:

increase (a stored value) **by** (value)

This phrase increases the variable, table entry, list entry, or property by the given amount, which must be of a compatible kind. Example:

```
increase the score by 8;
increase the time of day by 5 minutes;
```

decrease (a stored value) **by** (value)

This phrase decreases the variable, table entry, list entry, or property by the given amount, which must be of a compatible kind. Example:

```
decrease the score by 6;
decrease the carrying capacity of the player by 10;
```

An even greater abbreviation can be made when we are changing a number by 1 either way:

increment (a stored value)

This phrase increases the variable, table entry, list entry, or property by 1. Example:

```
increment the score;
```

decrement (a stored value)

This phrase decreases the variable, table entry, list entry, or property by 1. Example:

```
decrement the score;
```

"Increment" and "decrement" are traditional computing terms, though they have been used in engineering for at least a century and in finance for longer still.

Start of Chapter 8: Change
Back to §8.11. Now...
Onward to §8.13. Checking on whereabouts

§8.13. Checking on whereabouts

We have seen that while rooms are fixed, their contents move around, so we will need ways to examine the current whereabouts of things. The following examples show the kind of conditions allowed:

```
if the genie's lamp is in Aladdin's Cave ...
if Aladdin is not in Aladdin's Cave ...
```

```
if Aladdin's Cave contains the genie's lamp ...
if the genie's lamp is carried by Aladdin ...
if Aladdin is carrying the genie's lamp ...
if Aladdin does not have the genie's lamp ...
if the table supports the genie's lamp ...
if the table is supporting the genie's lamp ...
if the genie's lamp is supported by the table ...
if the genie's lamp is on the table ...
if the genie's lamp is on top of the table ...
if the genie's lamp is in the cupboard ...
if the genie's lamp is contained in the cupboard ...
if the genie's lamp is inside the cupboard ...
if the genie's lamp is within the cupboard ...
if the wick is part of the genie's lamp ...
```

These are exactly like the assertions which we use to set up the world, except that we make them questions by placing "if" in front. But we shall later see that we can also use three other tenses, not to mention plural forms, so that new verbal forms like "had not been inside" and "were not supported by" are legal here (which they would not be in assertions). What we are not allowed is to contract these verbs with apostrophes: "isn't", "hasn't" and "hadn't" are forbidden.

Overwhelmingly the condition we check most is whether the player is carrying something. The following are therefore equivalent:

```
if the genie's lamp is carried by the player ...
if the genie's lamp is carried ...
```

And similarly for "not carried", "worn" and "not worn". To be precise, if a form of *to be carried* or *to be worn* is not followed by any other description, then "the player" is assumed to be doing the carrying or wearing.

Start of Chapter 8: Change

Back to §8.12. Increasing and decreasing

Onward to §8.14. More flexible descriptions of whereabouts

§8.14. More flexible descriptions of whereabouts

The examples just given were all basically of the form "X *relation* Y" where X and Y were specific names of things. For example,

```
if the genie's lamp is carried by Cinderella ...
if the genie's lamp is inside the cupboard ...
```

Just as actions could be described with patterns to be matched ("taking an open container", say), so can the positions of things. Giving subtler descriptions of our X and Y sometimes broadens the possibilities, sometimes narrows them:

```
if the genie's lamp is carried by a woman ...
if the genie's lamp is inside the closed cupboard ...
```

In the first case, Y is allowed to be one of a whole range of things - any of the women existing in the world. This makes for a broader condition. In the second case, Y has not only to be the cupboard, but at a time when it is closed: which makes for a narrower condition. We can, of course, also vary X:

```
if an animal is inside the cupboard ...
if a container is carried ...
```

And we can even vary both X and Y at once:

```
if a woman is holding an animal ...
```

a condition which will be true if, anywhere in the story's world, any woman is holding any animal.

Start of Chapter 8: Change

Back to §8.13. Checking on whereabouts

Onward to §8.15. Calling names

§8.15. Calling names

Conditions like "if somebody is in an adjacent room" allow complicated tests to be performed with a minimum of fuss, but it's rare that we want to know only whether they are true: more likely we also want to know *which* person, and *which* room.

For this purpose, we are allowed to supply a name for any such vaguely-described object which comes up, and then to use that name thereafter.

```
if somebody is in an adjacent room (called the Hiding Place), say
"You hear distant breathing from [the Hiding Place]."
```

We can even name more than one of the things discovered:

```
Instead of waiting when a woman (called the kidnapper) is holding an
animal (called the pet), say "How can you think of rest when,
somewhere out there, [pet] has been cruelly kidnapped by [the
kidnapper]?"
```

Note the brackets, which are essential. The result of typing "wait" is then

```
How can you think of rest when, somewhere out there, a lapdog has
been cruelly kidnapped by Baroness Orczy?
```

Of course, that might be just one of many animals held by women in the story. We shall later see ways to go through all of the possibilities found, performing some action with each in turn.

A calling, if we can use that word, should be made immediately after the noun it refers to, and not left to hang back after any relative clauses. For instance,

```
if something (called the penitential object) held by the player is hot
```

is allowed, but not

```
if something held by the player (called the penitential object) is hot
```

because there is too much potential ambiguity - are we trying to call the player something?

See **Repeat running through** *for systematically working on everything matching a description*

Start of Chapter 8: Change

Back to §8.14. More flexible descriptions of whereabouts
Onward to §8.16. Counting the number of things

Example 128: **Higher Calling** All doors in the game automatically attempt to open if the player approaches them when they are closed.

---

§8.16. Counting the number of things

It is very often useful to know how many things are in a given situation, and for this purpose we have the "number of ..." construction. For instance:

```
the number of edible things carried
the number of things on the table
the number of people in the Dining Room
```

Whereas "a woman is holding an animal" makes the same test as "an animal is held by a woman", getting the same result, counting is not so even-handed:

```
the number of women holding animals
the number of animals held by women
```

are different questions and, unless the ration is strictly one lapdog per baroness, will have different answers. If Cruella de Vil has 101 dalmatians, they may be very different indeed.

It can also be helpful to count things with no particular location, like so:

```
the number of rooms
the number of closed doors
```

For instance:

```
When play begins, change the right hand status line to "Explored:
[number of visited rooms]/[number of rooms]".
```

Provided that the possible range is finite, we can also use "number of" to count values which match a description. For instance:

```
the number of non-recurring scenes
```

or if we were to define

```
Colour is a kind of value. The colours are red, orange, yellow, green,
blue, indigo and violet.
```

then "the number of colours" would evaluate to 7. As with other ways of talking about whole ranges of values, this only works if the range is manageable. "The number of numbers" cannot sensibly be worked out: there are infinitely many, for all practical purposes, and similarly for "the number of texts".

Start of Chapter 8: Change

Back to §8.15. Calling names

Onward to §8.17. Looking at containment by hand

§8.17. Looking at containment by hand

The descriptions outlined in the last few sections are intended to deal with almost all of the routine questions we might have about what currently resides where. It should be a last resort to use the following more primitive way to inspect the world.

holder of (object) **... object**

This phrase produces the container, supporter, carrier, wearer or room in which the object resides.

It's sometimes useful to go the other way. When something has possessions, we can find them out one at a time by running through a list.

first thing held by (object) **... object**

This phrase produces the first of the list of things held by the object. Example:

```
first thing held by Baroness Orczy
```

next thing held after (object) **... object**

This phrase produces the next item of the list of things held by something. Example: suppose Baroness Orczy is carrying a lapdog and a string of pearls.

```
next thing held after the lapdog
```

is then the string of pearls.

Start of Chapter 8: Change
Back to §8.16. Counting the number of things
Onward to §8.18. Randomness

§8.18. Randomness

Sometimes we want to introduce random behaviour into play. We usually do this by generating random values, and then acting differently depending on what they are. The following:

```
a random number from 2 to 5
```

produces, as it suggests, a random number drawn from the choices 2, 3, 4 or 5, each of which is equally likely to come up. In fact, this isn't limited to numbers:

a random (name of kind) **between** (arithmetic value) **and** (arithmetic value) **... value**

or:
a random (name of kind) **from** (arithmetic value) **to** (arithmetic value) **... value**

or:
a random (name of kind) **between** (enumerated value) **and** (enumerated value) **... value**

or:
a random (name of kind) **from** (enumerated value) **to** (enumerated value) **... value**

This phrase produces a uniformly random value in the range given. Examples:

```
a random number from 10 to 99
a random time from 2:31 PM to 2:57 PM
```

If we make a new kind of value:

```
A cloud pattern is a kind of value. The cloud patterns are
cumulus, altocumulus, cumulonimbus, stratus, cirrus,
nimbus, nimbostratus.
```

then we can also take random values from it:

```
a random cloud pattern between stratus and nimbus
```

which has three possible outcomes, all equally likely.

We can also use random conditions:

if a random chance of (number) **in** (number) **succeeds:**

This condition is true X/Yths of the time, where X and Y are the numbers.

Example:

```
if a random chance of 2 in 3 succeeds, ...
```

Here is a rule which applies only 15% of the time:

```
Instead of waiting when a random chance of 15 in 100
succeeds: ...
```

Testing IF which makes random choices can be rather frustrating, because a problem showing up on one attempt may not show up on another. We can get around this by making use of the fact that computers do not actually generate true randomness, but instead make a sequence of apparently random numbers by applying a complicated formula to each one in order to make the next. The starting point is a number called the "seed", because the next choice grows out of it.

seed the random-number generator with (number)

This phrase changes the seed number as specified. Any random numbers generated after that depend only on the seed. Example: the following sentence will "fix" the process of generating these random numbers so that they are not random at all - the same sequence of random numbers will be produced on each run.

```
When play begins, seed the random-number generator with
1234.
```

The seed value "1234" can be anything positive; a different sequence of random numbers will be produced for each different seed value. A seed value of 0 restores the RNG to properly random behaviour again.

Alternatively, it's possible the "fix" the RNG by clicking the "Make random outcomes predictable when testing" option on the Settings panel. This makes the behaviour predictable whenever the story is played within Inform, but (unlike the rule above) has no effect on the story file once released.

Start of Chapter 8: Change
Back to §8.17. Looking at

containment by hand

Onward to §8.19. Random choices of things

Example 129: ★ **Do Pass Go** A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

Example 130: ★ **Lanista 1** Very simple randomized combat in which characters hit one another for a randomized amount of damage.

Example 131: ★ **Weathering** The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

Example 132: ★★★ **Uptown Girls** A stream of random pedestrians who go by the player.

---

§8.19. Random choices of things

Writing "a random number" is not allowed, because the possible range is too large, but that was the only reason why not.

a/-- **random** (description of values) **... value**

This phrase makes a uniformly random choice from values satisfying the description given. Example:

```
a random visited room
a random scene
```

A problem message is issued if the range is too large (for instance, "a

random text"). Unexpected results may follow if no value fits the description, unless we are describing objects, in which case the result is the special value "nothing".

For instance:

```
say "You can see [number of adjacent rooms] way[s] from here; how about [random adjacent room]?"
```

But it's important to worry about the possibility that nothing qualifies - here, that no adjacent rooms exist. The above would then say:

```
You can see 0 ways from here; how about nothing?
```

Start of Chapter 8: Change

Back to §8.18. Randomness

Onward to Chapter 9: Time: §9.1. When play begins

Example 133: ★ **Candy** One of several identical candies chosen at the start of play to be poisonous.

Example 134: ★ **Zork II** A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby.

Examples from Chapter 8: Change

Start of this chapter

Chapter 9: Time

Indexes of the examples

Example Don Pedro's Revenge

RB

Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

Suppose our game features a detailed simulated combat between the player character and his opponent. He might have several weapons available, and several types of attack available; and at any given time he might be perched up in the rigging of his ship, standing on the open deck, or boxed in between some barrels. His options will vary depending on his position, and obviously it would detract from the pacing to make the player keep LOOKing in the middle of combat in order to remind himself where he is. Instead, we'll roll this information into the command prompt:

```
"Don Pedro's Revenge"

The Deck of the Helene Marie is a room. "The two
crews are embattled all around you, but your attention is
reserved for your particular enemy: Don Pedro."

Table of Random Prompts

position
prompt
"So securely boxed-in that you can really only parry or thrust, you
try to "
"Trapped between your barrels, you decide to "
"Boxed in, you choose to "
"Perched up here with the advantage of height (but little mobility),
you attempt to "
"Out on the open deck with no impediments, free to advance or
retreat, you decide to "

When play begins: reset the prompt.

Every turn: reset the prompt.
```

```
To reset the prompt:
    sort the Table of Random Prompts in random order;
    repeat through the Table of Random Prompts:
        if the position entry is the placement of the player:
            now the command prompt is prompt entry;
            stop.
```

```
After reading a command: say conditional paragraph
break.
```

```
A placement is a kind of value. The placements are
boxed, perched, free. The player has a placement. The
player is free.
```

```
Understand "retreat" or "parry" as retreating. Retreating
is an action applying to nothing.
```

```
Check retreating:
    if the player is perched, say "You can't move
backward or parry very successfully from this position."
instead.
```

```
Carry out retreating:
    now the player is boxed;
    say "You protect yourself, but end up wedged in
between two barrels."
```

```
Understand "thrust" or "advance" as advancing.
Advancing is an action applying to nothing.
```

```
Check advancing:
    if the player is perched, say "You can't move forward
```

```
from here, only slash." instead.


Carry out advancing:
    now the player is free;
    say "You push forward aggressively, making your
way to the open deck."


Instead of jumping:
    now the player is perched;
    say "You leap and swing yourself boldly up into the
rigging, leaving your attackers beneath you."


Instead of jumping when the player is perched:
    now the player is free;
    say "You leap down from your position, into the
middle of the deck."


Test me with "advance / jump / advance / retreat / jump
/ retreat / retreat / advance".
```

Of course, this won't be much fun until we also provide the player with a few weapons, some more fighting maneuvers, and, most of all, a Don Pedro to defeat.

---

114 Example Politics as Usual *RB*

Have the status line indicate the current region of the map.

Suppose a game with a large map entirely subdivided into regions. We could define:

```
"Politics as Usual"
```

```
When play begins:
    now the right hand status line is "[map region of the
location]".



Washington is west of Idaho.



Red is a region. Blue is a region. Idaho is in red.
Washington is in blue.



Test me with "e / w".
```

Note that, since regions can be stacked, we technically can be within more than one region at once. In the Port Royal example, for instance, the Tavern region is inside the Inland region. If there is any ambiguity, "the map region of the location" will be construed as "the smallest region that the location belongs to": so we would see "Tavern" rather than "Inland" in the status bar, when the player was in the Feathers or the Feathers Bedroom.

Some extra finesse would be necessary if the names of map regions were very long or if there were some rooms that were not considered to belong to any region at all.

---

115 **ExampleCentered** *RB*

Replacing the two-part status line with one that centers only the room name at the top of the screen.

If we want to lay out the status line in some other way than with left-hand and right-hand entries, it is possible to do this as well. Later we will learn about the "rule for constructing the status line", but here is a basic effect using this rule and an Inform extension included as part of the standard distribution, called Basic Screen Effects.

```
"Centered"
```

```
When play begins:
    say "After months of boring through the Earth's crust
in this metal-jawed vehicle, you break through..."


The Hollow Core is a room. "Truly a magnificent sight:
the land curves up away from you in every direction,
covered with the cities and fields of the Core People.
Molten rock runs in the canals, bringing heat and light to
every home.


At the center of the Earth hangs a dense black sun."


Include Basic Screen Effects by Emily Short.


Rule for constructing the status line:
    center "[location]" at row 1;
    rule succeeds.


Test me with "look".
```

Basic Screen Effects also provides a mechanism for building complicated status lines of more than one row. To read its documentation, we include the extension, press Go!, and then consult the contents index that results.

ExampleVitrine *RB*

An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

```
"Vitrine"

Plaza View is a room. "Your uncle's apartment, on loan
to you for viewing the parade and celebrations today.
This would be more of a sacrifice on his part if he
weren't currently yachting around Corfu."

The smart window is a device in Plaza View. It is fixed in
place. "A vast smart window [if transparent]overlooks
the park[otherwise]has turned to a sheet of hazy
blue[end if]." The smart window can be transparent. The
smart window is transparent. The description is "An
electrochromic device which changes shade and
transparency in response to the application of current.

Curtains are so last year."
```

Note the "can be transparent" line. Devices ordinarily are not allowed to have transparency or opaqueness, but we can make an exception in this case. Without that line, attempts to change the transparency of the window will fail.

```
Carry out switching off the window: now the window is
transparent.

Carry out switching on the window: now the window is
opaque.

Instead of searching a transparent window: say "Isn't it
lovely out there?"

Instead of searching an opaque window: say "The
window is currently darkened."
```

Test me with "look through window / switch window / look through window / look".

---

117 **Example Thirst** *RB*

A waterskin that is depleted as the player drinks from it.

"Thirst"

The player carries a waterskin. The waterskin can be full, partly drained, or empty. The waterskin is full. Understand "water" as the waterskin.

Instead of drinking the waterskin when the waterskin is empty:
    say "There is no water left."

Instead of drinking the waterskin: if the waterskin is partly drained, now the waterskin is empty; if the waterskin is full, now the waterskin is partly drained; say "You drink a long draught."

After printing the name of the waterskin: say " ([waterskin condition])"

Campsite is a room. "It is solid night now, and the stars have come out. Unfamiliar stars. On the other side of the valley -- a valley round-bottomed but shallow, like a soup bowl -- burn other campfires, most likely bandits. Their voices do not carry, but the smoke rises and

obscures the starlight over that way."

A sleepsack is an enterable container in the Campsite. "Your sleepsack is laid out in a pocket of sandy soil and coarse grass."

The sandy soil, the stars, the distant campfires, and the coarse grass are scenery in the Campsite. Understand "smoke" as the campfires. Instead of listening in the presence of your campfire: say "All you hear are the reassuring snaps and cracks of the sticks in your fire." Understand "campfires" or "fires" as the distant campfires.

Your campfire is scenery in the Campsite. Instead of pushing, pulling, turning, tasting, or touching your campfire, say "You would burn yourself." Understand "fire" as your campfire. The description of your campfire is "A reassuring protection against wild animals and cold."

The description of the stars is "You invent constellations for them. The slingshot. The scroll. The heart (upside down)."

Instead of going nowhere when the player is in Campsite:
    say "Now is not the time for wandering, alone in the dark. Better to keep here[if your campfire is visible], by the fire[end if]."

Singing is an action applying to nothing. Understand "sing" as singing.

Instead of singing:
    say "You sing, deep and low, a song from home. It is a good night for singing and the song raises your spirits."

Test me with "i / drink water / i / drink water / i".

---

118 **Example Thirst 2** *RB*

A campfire added to the camp site, which can be lit using tinder.

"Thirst"

The player carries a waterskin. The waterskin can be full, partly drained, or empty. The waterskin is full. Understand "water" as the waterskin.

Instead of drinking the waterskin when the waterskin is empty:
    say "There is no water left."

Instead of drinking the waterskin: if the waterskin is partly drained, now the waterskin is empty; if the waterskin is full, now the waterskin is partly drained; say "You drink a long draught."

After printing the name of the waterskin: say " ([waterskin condition])"

Campsite is a room. "It is solid night now, and the stars have come out. Unfamiliar stars. On the other side of the valley -- a valley round-bottomed but shallow, like a soup bowl -- burn other campfires, most likely bandits. Their voices do not carry, but the smoke rises and obscures the starlight over that way."

A sleepsack is an enterable container in the Campsite. "Your sleepsack is laid out in a pocket of sandy soil and coarse grass."

The sandy soil, the stars, the distant campfires, and the coarse grass are scenery in the Campsite. Understand "smoke" as the campfires. Instead of listening in the presence of your campfire: say "All you hear are the reassuring snaps and cracks of the sticks in your fire." Understand "campfires" or "fires" as the distant campfires.

Your campfire is scenery in the Campsite. Instead of pushing, pulling, turning, tasting, or touching your campfire, say "You would burn yourself." Understand "fire" as your campfire. The description of your campfire is "A reassuring protection against wild animals and cold."

The description of the stars is "You invent constellations for them. The slingshot. The scroll. The heart (upside down)."

Instead of going nowhere when the player is in Campsite:
    say "Now is not the time for wandering, alone in the dark. Better to keep here[if your campfire is visible], by the fire[end if]."

Singing is an action applying to nothing. Understand "sing" as singing.

Instead of singing:
    say "You sing, deep and low, a song from home. It is a good night for singing and the song raises your spirits."

The player carries a tinderbox. The tinderbox contains a flint, a steel, some tinder, and a patch of carbonized cloth. The description of the flint is "A flat grey stone with flaked edges." The description of the steel is "Curved so that you can hold it over the knuckles of your right hand and strike it against the flint. There is a knack to it. Those without the knack end up with bloody knuckles and no fire." The steel is wearable. The description of the tinder is "Dried grass and similar." The description of the cloth is "The little, precious, spark-preserving scraps without which the fire would never begin."

Instead of attacking the flint when the steel is not worn by the player:
    say "You must wear the steel over your knuckles, in order to hit the flint at the best angle."

Instead of attacking the flint when the steel is worn and the cloth is not in the location:
    say "Though you strike the flint sharply with the steel and throw sparks, they have nothing to catch on, since the patch of cloth does not lie beneath."

Instead of attacking the flint when the steel is worn and the cloth is in the location and the cloth is not unlit:
    say "The patch of cloth has already caught."

```
Instead of attacking the flint:
    now the cloth is glowing;
    say "You strike the flint against the steel and throw
sparks onto the patch of cloth; they make tiny circles of
orange there, which will only prosper if blown into
flame."
```

Realistically, we ought to attach a randomization to this so that each step of the fire-starting has a good chance of failure. But because our player may not be as patient as someone who actually needs a fire started, we allow him to succeed the first time in every case.

```
Ignition is a kind of value. The ignitions are whole,
fading, glowing, flaming. A thing has an ignition.
```

```
Blowing on is an action applying to one thing.
Understand "breathe on [something]" or "blow on
[something]" as blowing on.
```

```
Instead of blowing on the whole cloth:
    say "There is no point, since no sparks have caught
there."
```

```
Instead of blowing on the fading cloth:
    now the cloth is flaming;
    say "You blow on the faint sparks on the cloth and
turn them into the beginnings of flame."
```

```
Instead of burning the whole tinder:
    if the cloth is not flaming:
        say "The patch of cloth must catch flame before
you can light anything with it.";
    otherwise:
        now the tinder is flaming;
        now the cloth is nowhere;
        say "You light the tinder with the patch of cloth, and
have the elements of a fire."
```

```
Every turn:
    unless the cloth is flaming or the cloth is whole:
        now the ignition of the cloth is the ignition before
the ignition of the cloth;
        say "Now the patch of cloth is [ignition of the
cloth]."



Some kindling is in the campsite.



Instead of burning the whole kindling:
    if the tinder is not flaming:
        say "You need the tinder to be flaming, first.";
    otherwise:
        now the tinder is nowhere;
        now the kindling is nowhere;
        move the campfire to the location;
        say "You succeed in lighting yourself a proper
campfire.";
        now the printed name of Campsite is "By The
Campfire".



Test me with "i / drink water / i / drink water / i / wear
steel / get flint / get cloth / drop cloth / get tinder / hit flint
/ blow on cloth / burn tinder / burn kindling / look".
```

ExampleMeteoric I and II *RB*

A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

The game below begins at half past eleven, and one turn later, it's meteor time:

```
"Meteoric I"
```

```
The time of day is 11:30 PM.


At 11:31 PM:
    now the meteor is in the great outdoors;
    say "A meteor streaks across the sky.".


The great outdoors is a region. The Spanish Balcony is
east of the Inner Court. The Court and Balcony are in
the great outdoors. Inside from the Court is the
Swimming Pool.


The meteor is a backdrop. Instead of doing something
to the meteor, say "The meteor is no longer visible, now
nothing more than a memory."


Test me with "wait / wait / examine meteor / west /
examine meteor / in / examine meteor".
```

Or for something a little slower-moving and with no after-image:

```
"Meteoric II"


The time of day is 4:30 PM.


At 4:31 PM:
    now Phobos is in the great outdoors;
    say "Phobos rises from the western horizon."
At 10:06 PM:
    now Phobos is nowhere;
    say "Phobos sets over the eastern horizon."
```

```
The great outdoors is a region. The Martian Balcony is
east of the Inner Court. The Court and Balcony are in
the great outdoors. Inside from the Court is the Heavy
Water Swimming Pool.
```

```
Phobos is a backdrop. Instead of doing something to
Phobos, say "Phobos orbits a mere 6000km above you,
which is practically touching range for astronomy. On
the other hand, astronomy isn't all that practical."
```

```
Test me with "wait / wait / examine phobos / west /
examine phobos / in / examine phobos".
```

Though we should not really use Earthly time-keeping, since the Martian day is about half an hour longer than ours.

---

120 Example Orange Cones *RB*

Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

Because we can invent our own adjectives (see the chapter on Phrases), we can make the conditions for a backdrop as simple or as complicated as we like.

In this scenario, we want the player to be able to take, move, and drop orange traffic cones to seal off one street or another. So we create our own "accessible" adjective as follows:

```
Definition: a road is accessible if the orange cones are
not in it.
```

...and now

```
move the traffic backdrop to all accessible roads.
```

will tell the traffic backdrop where to appear.

"Orange Cones"

A road is a kind of room.

The traffic is a backdrop. It is not scenery. The initial appearance is "Dense traffic snarls the streets, making it difficult to cross even with the lights. Men on motorbikes edge between the cars, and sometimes pull up onto the sidewalks to go around." The description is "It is more or less as usual for this time of day. It's a wonder it ever dissipates, really."

When play begins:
    move the traffic backdrop to all accessible roads.

A line of orange cones are a thing.

Definition: a road is accessible if the orange cones are not in it.

After dropping the orange cones in a road:
    say "With steely determination you begin to lay out the orange cones, blocking access to this segment of street. This produces honking and swearing -- but you persevere.";
    update backdrop positions.

After taking the orange cones:
    say "You go around taking up the orange cones, and within moments the traffic begins to flow into the street

```
again.";
    update backdrop positions.
```

That accomplishes everything we set out to do, but let's add a very simple puzzle to test it out with:

```
The Town Square is a road. North of the Town Square
is Candle Street. Candle Street is a road. East of Town
Square is Mortar Street. Mortar Street is a road.

The line of orange cones are in Mortar Street.

The player wears a reflective vest and a hard hat.

The manhole cover is a door. "[if location is
accessible]Under the cars in the middle of the street,
you can just make out the cover of the manhole you
need to get into.[otherwise]There's a promising manhole
in the middle of the street.[end if]". It is closed and
openable. It is below the Town Square and above the
Access Tunnel.

Instead of opening the manhole cover when the location
is an accessible road:
    say "You can't get anywhere near the manhole cover
with all these cars above it."

Instead of taking the orange cones in Town Square:
    if the manhole cover is closed or Town Square is
accessible, continue the action;
    otherwise say "You'd better not let the traffic back in
until you've closed the manhole. There'll be accidents
otherwise."
```

```
After going to the Access Tunnel:
    say "With a stealthy glance left and right, you lower
yourself into the access tunnel, thus accomplishing your
mission for Chapter 2. To continue your adventure, see
Chapter 3: The Vault of Peaquod.";
    end the story finally.
```

And finally, a couple of features from the Activities chapter to make the output more elegant:

```
Rule for writing a paragraph about the orange cones:
say "A line of orange cones holds back the traffic from
entering here."
```

```
Rule for printing the name of the orange cones when the
cones are carried by the player:
    say "stack of orange cones".
```

```
Test me with "x traffic / open manhole / n / x traffic / s / e
/ x traffic / get cones / look / x traffic / w / drop cones /
look / open manhole / take cones / d".
```

---

121 **Example** **Terror of the Sierra Madre** *RB*

Multiple player characters who take turns controlling the action.

Suppose we have a game where we want the player to control two different characters, swapping bodies from one turn to the next. First, the setting, and the two people who will alternately play:

```
"Terror of the Sierra Madre"
```

```
The Hay-Strewn Corridor is a room. "[if the player is
Maleska]The horse stalls are empty: you have already
```

```
drained the animals, and carried off their corpses. The
house will not long sustain you now.
```

```
The window throws on the floor a bright square of
malevolent sunlight[otherwise]The stalls for horses run
down one side of the room, but the house has long
stood empty. A square window without shutters looks
out over the ranch, away toward the Sierras[end if]."
```

```
Teresa is a woman in the Hay-Strewn Corridor. "Teresa
stands opposite you[if Teresa carries something], her
fingers wrapped tightly around [a list of things carried by
Teresa][end if]." Teresa carries a bulb of garlic and a
cross.
```

```
Maleska is a man in the Hay-Strewn Corridor. "Maleska
watches you from eyes entirely black." Maleska carries
a skull.
```

If we tried the text above in Inform, we would find ourselves in the Hay-Strewn Corridor and confronted by both Teresa and Maleska. If "player" is not set to any named person, Inform creates a bland person called "yourself" to represent the player. To avoid this, we set "player" to the person we want to begin as. The player character is normally privately-named, so we'll need to make sure "Maleska" still means what it should.

```
The player is Maleska. Understand "Maleska" as
Maleska.
```

Now the Corridor contains just two people, and we arrive on the scene as Maleska, with only Teresa facing us.

At the end of every turn we will use the 'now the player is...' phrase. (This looks as if it simply changes the value of "player": which it does, but it also carries out a complicated operation behind the scenes to effect the switch.)

```
Every turn:
    if the player is Maleska, now the player is Teresa;
```

```
        otherwise now the player is Maleska.
```

Our two characters already see the Corridor differently, but let's differentiate them further:

```
Every person has a number called strength. The
strength of Teresa is 3. The strength of Maleska is 5.
```

In this small example, strength is not used for anything, except that we will display it on the status line:

```
When play begins:
    now the command prompt is "[bold type][player]
[roman type] > ";
    now the left hand status line is "[player]";
    now the right hand status line is "STR: [strength of
the player]".
```

That last rule doesn't quite do what we might have expected. When we print "[player]", we find that Inform usually prints "yourself". This is because Inform says "you" to mean Teresa when talking to the player-being-Teresa, and likewise for Maleska. We want to override that in this particular story, because the rapid switches of personality are otherwise hard to follow. So:

```
Rule for printing the name of Teresa: say "Teresa".

Rule for printing the name of Maleska: say "Maleska".

Test me with "look / look".
```

Some kinds of game objects -- food, for instance -- can only sensibly be used once, and should then be destroyed. The EAT command already implements this, but suppose we also had a category of drinkable potions:

```
"Beverage Service"

A potion is a kind of thing. The sparkly blue potion is a
potion carried by the player.

Level 3 is a room.

Instead of drinking a potion (called the drink):
    now the drink is nowhere;
    say "You quaff [the drink]. It goes down beautifully."

Test me with "drink sparkly / i".
```

---

123 **Example** **Spring Cleaning** *RB*

A character who sulks over objects that the player has broken (and which are now off-stage).

Here we have a destruction action that allows the player to break any fragile items. Once destroyed, these things are removed from play, but we can still refer to them: they are now off-stage. This makes it easy for our sulking character to list the ones that have been destroyed:

```
"Spring Cleaning"
```

A thing can be tough or fragile. A thing is usually tough.

Instead of attacking something fragile:
    say "You smash [the noun] to smithereens!";
    now the noun is nowhere.

A knick-knack is a kind of thing which is fragile.

Every turn when a knick-knack is off-stage and Granny Blue can see the player:
    say "'Ohh,' whimpers Granny to herself softly. 'How I will miss [the list of off-stage knick-knacks]!'"

The Parlor is a room. Granny Blue is a woman in the Parlor. A china lamb, a porcelain milkmaid, a frolicking Dutch cow, and a crystal unicorn are knick-knacks in the Parlor.

Test me with "break lamb / break milkmaid / break cow / break unicorn".

ExampleExtra Supplies *RB*

A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

Suppose we have a supply closet in our game from which the player is allowed to take red pens. To keep modeling simple, we only allow him to have one in play at a time, and we test this by seeing whether the red pen is "off-stage" before moving it to his possession.

This approach might seem no different from having a single red pen sitting in the closet, but it may be preferable, for two reasons. First, it's not very plausible for a supply closet to contain nothing but a single red

pen (well, assuming a well-regulated supplier, anyway); and second, it gives the player a way to get a new red pen should the original be destroyed in a tragic handwriting accident.

```
"Extra Supplies"


The Supply Closet is a room. A supply of red pens is in
the Supply Closet. Understand "pen" as the supply of
red pens when the red pen is not visible.


There is a red pen.


Instead of taking the supply of red pens:
    if the red pen is off-stage:
        move the red pen to the player;
        say "You help yourself to a fresh red pen.";
    otherwise:
        say "You're only allowed one pen at a time. The
department secretary is very strict."


South of the Supply Closet is the Furnace Room. The
incinerator is a thing in the Furnace Room. It is a
container. "The incinerator is here, working full blast."


After inserting something into the incinerator:
    now the noun is nowhere;
    say "A fiery blast consumes [the noun]!"


Test me with "get pen / i / get pen / get supply / s / put
pen in incinerator / n / get pen".
```

Mazes are a traditional element of interactive fiction, often consisting of apparently identical rooms with exits that do not work reciprocally and which cause confusion.

The methods of mapping mazes are now fairly well understood and mazes themselves tend to be regarded as tiresome rather than enjoyable by a large portion of the playing audience. However, if we did want to ignore the common wisdom and create a maze, randomly generated at the start of play, here would be one way to go about it:

```
"Maze of Gloom"


A Bee Chamber is a kind of room. The printed name of
a Bee Chamber is usually "Hexagonal Room". The
description of a Bee Chamber is usually "Waxy,
translucent walls surround you on six sides; the floor
and ceiling are made of the same material, gently
uneven. There are exits in every direction, cut into the
faces or the corners."


Bee1, Bee2, Bee3, Bee4, Bee5, Bee6, Bee7, Bee8,
Bee9, and Bee10 are Bee Chambers.


When play begins:
    now right hand status line is "[number of visited
rooms]/[number of rooms]";
    repeat with place running through Bee Chambers:
        now a random Bee Chamber is mapped north of
place;
        now a random Bee Chamber is mapped northwest
of place;
        now a random Bee Chamber is mapped west of
place;
        now a random Bee Chamber is mapped southwest
of place;
```

```
        now a random Bee Chamber is mapped south of
place;
        now a random Bee Chamber is mapped southeast
of place;
        now a random Bee Chamber is mapped east of
place;
        now a random Bee Chamber is mapped northeast
of place;
        now a random Bee Chamber is mapped above
place;
        now a random Bee Chamber is mapped below
place;
        now a random Bee Chamber is mapped inside
place;
        now a random Bee Chamber is mapped outside
place.



Test me with "in / out / up / down / n / ne / nw / e / w / sw
/ se / s".
```

ExampleHatless

RB

It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

Suppose we want a game in which each scenario starts with the characters wearing hats -- randomly passed out. We might be tempted to write our scenario like this:

```
"Hatless"



The Costumery is a room. Larry, Curly, and Moe are
men in the Costumery. Janine is a woman in the
Costumery.



Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
```

```
    say "[The target] is here, looking stylish in [an attire]."

Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."

A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.

The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.

When play begins:
    now every hat is worn by a random hatless person.
```

And we might hope that this would choose a new hatless person for each hat, but we would be wrong. It will instead choose one hatless person and put all the hats on him -- and everyone else has to go bare-headed. That's clearly no good. Let's try again:

```
"Hatless 2"

The Costumery is a room. Larry, Curly, and Moe are
men in the Costumery. Janine is a woman in the
Costumery.

Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
    say "[The target] is here, looking stylish in [an attire]."
```

```
Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."
```

```
A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.
```

```
The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.
```

```
When play begins:
    now every hatless person wears a random hat.
```

But this selects one random hat and assigns it to each hatless person in turn -- so it will only wind up being worn by the last of them (since Inform knows that only one person can wear a hat at a time).

In this case, we do have to expand out our loop so that the game makes an explicit distribution:

```
"Hatless 3"
```

```
The Costumery is a room. Larry, Curly, and Moe are
men in the Costumery. Janine is a woman in the
Costumery.
```

```
Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
    say "[The target] is here, looking stylish in [an attire]."
```

```
Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."
```

```
A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.


The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.


When play begins:
    repeat with item running through hats:
        now the item is worn by a random hatless person.
```

Each time Inform considers the instruction "now the item is worn by a random hatless person", there is one fewer such person to choose from -- so we can guarantee that the hats are distributed one per customer and that all hats are distributed.

Hatless 3 is only guaranteed to work because the number of hats is less than or equal to the number of people; otherwise the final use of random will return "nothing" and then a problem message will appear during play.

Example **Technological Terror** *RB*

A ray gun which destroys objects, leaving their component parts behind.

```
"Technological Terror"


The Decomposition Ray Gun is a thing carried by the
player.
```

First we need to define our shooting action:

```
Shooting it with is an action applying to two things.
```

```
Check shooting something with something:
    if the player is not carrying the Ray Gun, say "You are
pathetically unarmed!" instead;
    if the second noun is not the Ray Gun, say "[The
second noun] does not fire." instead;
    if the noun is the Ray Gun, say "Nice trick if you can
do it!" instead;
    if the noun is the player, say "That would be
disastrous!" instead.
```

Next, some grammar to allow the player to use this action:

```
Understand "shoot [gun] at [something ungunlike]" as
shooting it with (with nouns reversed).
```

```
Definition: a thing is ungunlike if it is not the gun.
```

```
Understand "shoot [something ungunlike] with [gun]" as
shooting it with. Understand "shoot [something] with
[something]" as shooting it with.
```

```
Understand "shoot [something] at [something]" as
shooting it with (with nouns reversed). Understand "fire
[gun] at [something ungunlike]" as shooting it with (with
nouns reversed). Understand "fire at [something
ungunlike] with [gun]" as shooting it with. Understand
"fire at [something] with [something]" as shooting it with.
```

Strictly speaking, we only need these last grammar lines (with "understand shoot something...") in order to define an action that the player can take. Adding more grammar lines means that Inform will try to match the most specific ones first, which is useful when the player types something ambiguous and there is one choice that obviously fits this action better than the others. See the chapter on Understanding for a further discussion.

Here we get to use "now..." to give it its destructive effect:

```
Carry out shooting something with something:
    say "ZAP! [The noun] twinkles out of existence! [if
something is part of the noun][The list of things which
are part of the noun] clatter to the ground! [end if]
[paragraph break]";
    now every thing which is part of the noun is in the
location;
    now the noun is nowhere.


The Deathbot Assembly Line is a room. "Here is the
heart of the whole operation, where your opponents are
assembled fresh from scrap metal and bits of old car."
The dangerous robot is a thing in the Assembly Line.
"One dangerous robot looks ready to take you on!" A
robotic head, a drill arm, a needle arm, a crushing leg
and a kicking leg are parts of the dangerous robot.


Instead of examining something when something is part
of the noun:
    say "[The noun] consists of [a list of things which are
part of the noun]."


Test me with "x robot / shoot robot / fire at kicking leg /
shoot gun at drill arm / look".
```

Example **Higher Calling** *RB*

All doors in the game automatically attempt to open if the player approaches them when they are closed.

"Called" is quite useful in the context of rules about going, since go rules often refer to things other than the noun the player typed. For instance, suppose we want to have doors which politely open when the player tries to pass through them:

```
"Higher Calling"

Before going through a closed door (called the blocking
door):
    say "(first opening [the blocking door])[line break]";
    silently try opening the blocking door;
    if the blocking door is closed, stop the action.

Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.

The church door is east of Eastern End and west of the
Courtyard. The church door is a door.

Test me with "e / e".
```

A fuller set of such rules is included in the Locksmith extension.

Example **Do Pass Go** *RB*

A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

```
"Do Pass Go"
```

Go is a room. "A giant square area, where you and your other pewter ornament friends gather before setting out to purchase London."

The pair of dice is carried by the player.

The pair of dice has a number called first die. The pair of dice has a number called second die. The first die of the pair is 6. The second die of the pair is 6. Rule for printing the name of the pair of dice while taking inventory: say "pair of dice".
Rule for printing the name of the pair of dice: say "pair of dice showing [first die of the pair plus second die of the pair]".

To say detailed state of the dice:
    if the first die of the pair is the second die of the pair, say "double [first die of the pair]";
    otherwise say "[first die of the pair] and [second die of the pair]".

The description of the pair of dice is "The pair of dice are [if the dice are carried]itching to be rolled[otherwise]showing [detailed state of the dice][end if]."

Rolling is an action applying to one carried thing.
Understand "roll [something preferably held]" as rolling.

Check rolling when the noun is not the pair of dice: say "Not something you can roll." instead.
Carry out rolling:
    now the pair of dice is in the holder of the actor;
    now the first die of the pair of dice is a random number from 1 to 6;
    now the second die of the pair of dice is a random

```
number from 1 to 6.
Report rolling:
    say "You roll [detailed state of the dice]."
```

```
Test me with "i / roll dice / look / x dice / get dice / x dice
/ roll dice / roll dice / roll dice / roll dice / roll dice / roll
dice / roll dice".
```

Because we remember the states of the individual dice, not just a total, we can make use of the combination rolled.

```
The doubles count is a number that varies.
After rolling:
    if the first die of the pair is the second die of the pair,
increment the doubles count;
    otherwise now the doubles count is 0;
    continue the action.
```

```
Jail is a room. "This is Jail, and not the Just Visiting
periphery, either."
```

```
Every turn when the doubles count is 3:
    say "The blue-uniformed policemen blows his whistle
and beckons you sternly...";
    now the player carries the pair of dice;
    now the player is in Jail;
    now the doubles count is 0.
```

```
Every turn when the doubles count is 1 and the player is
in Jail:
    say "The warden gruffly releases you.";
    now the player carries the pair of dice;
    now the player is in Go.
```

ExampleLanista 1

Very simple randomized combat in which characters hit one another for a randomized amount of damage. *RB*

"Lanista, Part One"

The Arena is a room. "Sand, blood, iron. These festivals are normally held on hot days, but the sun has gone behind a cloud and fat drops of rain now and then spatter the arena floor." The gladiator is a man in the Arena. "A bare-chested Scythian gladiator faces you, wielding a trident."

We start by recording, for each person, a maximum number of points of damage the person can sustain when starting from health, and the current number of points remaining. In the tradition of role-playing games, these are referred to as hit points.

A person has a number called maximum hit points. A person has a number called current hit points.

The maximum hit points of the player is 35. The maximum hit points of the gladiator is 25.

The current hit points of the player is 35. The current hit points of the gladiator is 25.

Now our rule for the actual attack. We want first to calculate how much damage the player's attack does, inflict that damage, and remove the enemy if he's dead; then, if he doesn't die, the enemy counter-attacks, also for a randomized amount of damage, and if this kills the player, the game ends in defeat.

Instead of attacking someone:
    let the damage be a random number between 2 and

```
10;
    say "You attack [the noun], causing [damage] points
of damage!";
    decrease the current hit points of the noun by the
damage;
    if the current hit points of the noun is less than 0:
        say "[line break][The noun] expires, and is
immediately carried away by the Arena slaves!";
        now the noun is nowhere;
        end the story finally;
        stop the action;
    let the enemy damage be a random number between
2 and 10;
    say "[line break][The noun] attacks you, causing
[enemy damage] points of damage!";
    decrease the current hit points of the player by the
enemy damage;
    if the current hit points of the player is less than 0:
        say "[line break]You expire!";
        end the story.
```

This last bit is a refinement to help the player keep track of how the contest is going:

```
When play begins:
    now the left hand status line is "You: [current hit
points of player]";
    now the right hand status line is "Gladiator: [current
hit points of gladiator]".



Test me with "hit gladiator / g / g / g".
```

ExampleWeathering

RB

The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

```
"Weathering"
```

A cloud pattern is a kind of value. The cloud patterns are cumulus, altocumulus, cumulonimbus, stratus, cirrus, nimbus, nimbostratus.

The Mount Pisgah Station is a room. "The rocky peak of Mt. Pisgah (altitude 872m) is graced only by an automatic weather station. The clouds, close enough almost to touch, are [a random cloud pattern]. Temperature: [a random number from 7 to 17] degrees, barometric pressure: [950 + a random number from 0 to 15] millibars."

Test me with "look / look / look".

---

132 **Example** **Uptown Girls** *RB*

A stream of random pedestrians who go by the player.

Suppose we have an urban space we want to populate with random passers-by. These should have a range of characteristics and not always be described in the same way; and once the player has noticed one, he should be able to look at her further, until another pedestrian crosses his path.

"Uptown Girls"

Riverside Drive is a room. "There's a pleasant late-afternoon view of the Hudson, and a snap in the air, and you would rather be here than anywhere."

Instead of going a direction, say "Oh, you know where you're going; no need to deviate from the usual path."

```
Instead of waiting, say "You stroll along enjoying the
November crispness."
```

It gets a little annoying to have a random event occurring every single turn of play, so let's introduce some randomness to determine how often the message appears:

```
Every turn when a random chance of 1 in 3 succeeds:
    reset passerby;
    choose a random row in the Table of Atmospheric
Events;
    say "[event entry][paragraph break]"
```

```
Table of Atmospheric Events
event
"Slowly [a passerby] strolls by, turning to look at you as
she passes."
"Some [passerby] nearly bumps into you."
"You dodge to avoid [a passerby]."
"You weave around [a passerby], who has stalled to
look into a window."
"There's a ruckus as one of the ubiquitous taxis nearly
collides with [a passerby] crossing the street."
"[The passerby] beside you waves to a friend across the
street."
"To your left, [a passerby] drops her purse, and swears
as she retrieves it."
```

```
Hair color is a kind of value. A person has hair color. the
hair colors are red-headed, brunette, blonde.
```

```
Height is a kind of value. A person has height. The
heights are tall, medium-height, short.
```

```
Grooming is a kind of value. A person has grooming.
The groomings are messy and tidy.
```

```
To reset passerby:
    now the hair color of the passerby is a random hair
color;
    now the height of the passerby is a random height;
    now the grooming of the passerby is a random
grooming.
```

```
The passerby is a woman in Riverside Drive. The
passerby is scenery. Understand "woman" or "lady" as
the passerby. The printed name of the passerby is "[one
of]woman[or]lady[purely at random]".
```

At this point we borrow some techniques from later to describe the woman with a random combination of characteristics, and to let the player refer to her by those traits:

```
Before printing the name of the passerby:
    if a random chance of 1 in 4 succeeds, say "[height] ";
    if a random chance of 1 in 4 succeeds, say
"[grooming] ";
    if a random chance of 1 in 4 succeeds, say "[hair
color] ".
```

```
Understand the hair color property as describing the
passerby. Understand the height property as describing
the passerby. Understand the grooming property as
describing the passerby.
```

If we also wanted each of those combinations to mean some more specifically-described woman:

```
Instead of examining a passerby:
    repeat through Table of Passerby Descriptions:
        if hair entry is the hair color of the passerby and
height entry is the height of the passerby and grooming
entry is the grooming of the passerby, say "[description
entry][paragraph break]".
```

Table of Passerby Descriptions

[illegible]	[illegible]
[illegible]	"A tall-ish woman with long red hippie-hair poking out of a ponytail in straggles, and bent to hide how tall she is."
[illegible]	"A [illegible]-headed with shingled hair."
[illegible]	"A [illegible] urchin, and very young, with ginger hair and a smudged nose and far too many freckles."
[illegible]	"[illegible] career woman with henna-red hair."
[illegible]	"[illegible] in the way that lollipops and fire trucks are red: not by nature but by art. The rest of her clothing is pretty ordinary, though."
[illegible]	"[illegible] small in every sense, with chin-length red-hair. Even high heels do not bring her head much above your shoulder."
[illegible]	"[illegible]-esque woman with dark hair, wearing something resembling a tent."
[illegible]	"[illegible] woman with dark brown hair and the aura of needing a wash."
[illegible]	"There are mustard stains on the t-shirt of this short brown-haired woman. Estimated age ca. 40. Possibly homeless."
[illegible]	"[illegible] brunette in business attire."
[illegible]	"[illegible] height, brown-haired, generally nondescript."
[illegible]	"[illegible] little dark-haired girl."
[illegible]	"[illegible] blonde of about thirteen who looks as though she has not yet figured out how to get her wardrobe to catch up with her rate of growth. Her t-shirt and her pants are too short."
[illegible]	"[illegible] pants and the wall-o-hair look."
[illegible]	"One of those shocking platinum blonde types, with a tiger-patterned skirt. Reeally trashy."
[illegible]	"[illegible] and severe, with perfectly straight hair falling to the middle of the back."
[illegible]	"[illegible] Marilyn-esque blonde."
[illegible]	"[illegible] precise in a blue-and-white striped suit and a boyish haircut."

Test me with "z / z / x passerby / z / z / x passerby".

---

133 **ExampleCandy** *RB*

One of several identical candies chosen at the start of play to be poisonous.

Suppose we want to give the player a bag of candies, of which a random one is poisonous. We can pick which one should be poisoned at the start of play, like this:

"Candy"

The plural of piece of candy is pieces of candy. A piece of candy is a kind of thing. A piece of candy is always edible. Four pieces of candy are in the Halloween bag.

Toxicity is a kind of value. The toxicities are safe and poisonous. A piece of candy has a toxicity. A piece of candy is usually safe.

The Porch is a room. The player carries the Halloween bag.

After eating a poisonous piece of candy:
    say "Oh, that didn't taste right at all. Oh well!"

When play begins:
    now a random piece of candy is poisonous.

Test me with "eat candy / g / g / g".

ExampleZork II

A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby. *RB*

All we need to do is select the player's destination for him at random:

```
"Zork II"

The Carousel Room is a room.

Instead of going from the Carousel Room:
    move the player to a random adjacent room.
```

To avoid infringing the original game too much, let's try a somewhat different setting:

```
The Games of Chance is north of the Carousel Room.
The Haunted Funhouse is northwest of the Carousel
Room. The Ferris Wheel is east of the Carousel Room.
The Topsy-Turvy is northeast of the Carousel Room.
The Reproduction Henge is south of the Carousel
Room. The Women's Toilet is southwest of the Carousel
Room. The Men's Toilet is southeast of the Carousel
Room. The Cotton Candy Shop is west of the Carousel
Room.

Test me with "s".
```

And the following means that the test runs consistently even though the numbers are theoretically random. To make them truly random, remove this line.

```
When play begins, seed the random-number generator
with 1234.
```

Or if we want to add the refinement that the Carousel Room can be switched off:

```
"Zork II"
```

```
The Carousel Room is a room. The spinning machine is
a switched on device in the Carousel Room.
```

And then

```
Instead of going from the Carousel Room when the
spinning machine is switched on:
    move the player to a random adjacent room.
```

```
The Games of Chance is north of the Carousel Room.
The Haunted Funhouse is northwest of the Carousel
Room. The Ferris Wheel is east of the Carousel Room.
The Topsy-Turvy is northeast of the Carousel Room.
The Reproduction Henge is south of the Carousel
Room. The Women's Toilet is southwest of the Carousel
Room. The Men's Toilet is southeast of the Carousel
Room. The Cotton Candy Shop is west of the Carousel
Room.
```

```
Test me with "turn off machine / s / n / turn on machine /
s".
```

```
When play begins, seed the random-number generator
with 1234.
```

Chapter 9: Time

elling the time; *§9.8. Approximate times, lengths of time*; *§9.9. Comparing and shifting times*; *§9.10. Cal*

Contents of *Writing with Inform*
Chapter 8: Change
Chapter 10: Scenes
Indexes of the examples

§9.1. When play begins

With the material from previous chapters, we are now able to set up a simulated world, to respond to the player's actions within it, and to make it change in perhaps unexpected ways.

The resulting experience can be as lively as we want to make it, but so far we have no very good ways to give it any structure: a beginning and an end, for instance, or a sense of plot. This next chapter is all about the passage of time, and it begins at the beginning.

We have already seen an example of how to write a rule which applies just once, at the start of the story:

```
When play begins: say "Welcome to Old Marston Grange, a country
house cut off by fog."
```

The "when play begins" rules are checked only at the start of a story, not when a saved session is restored from disc. What happens is that these rules are followed, then the story's banner is printed, then the initial room description is printed up, and then the player is asked for a first command.

Start of Chapter 9: Time
Back to Chapter 8: Change: §8.19. Random choices of things
Onward to §9.2. Awarding points

Example 135: ★ **Clueless** A murderer for the mystery is

selected randomly at the beginning of the game.

§9.2. Awarding points

Traditionally-written stories award points throughout play, as an indication of progress. If we want to be traditional, we must first write:

```
Use scoring.
```

Without this, the SCORE, NOTIFY ON and NOTIFY OFF commands do not work; the final score is not shown at the end of a story; and the status line above the player's text area shows only the turn count, not (as is more usual) both the score and the turn count. Changing the "score" (see below) has no visible effect, though it is not actually illegal.

With "Use scoring" in place, we can award points as follows:

```
increase the score by 5;
```

substituting whatever number we feel is appropriate. We should be careful not to give out the same points over and over, that is, not to reward the same basic achievement many times over if the player simply repeats the same action. This, for instance, is open to abuse:

```
After taking the trophy:
    increase the score by 5;
    say "Well done!"
```

The player may simply take the trophy, drop it again, take it again, ... and win five points every time around. We can prevent this by phrasing the rule more carefully:

```
After taking the trophy when the trophy is not handled:
    increase the score by 5;
    say "Well done!"
```

Rather than being an open-ended scoring system, IF normally has a maximum possible score, which can be specified with a sentence like so:

```
The maximum score is 10.
```

The score and maximum score are just numbers that vary, so we can freely change them:

```
After eating the poisoned mushroom:
    now the score is -100.
```

Start of Chapter 9: Time

Back to §9.1. When play begins

Onward to §9.3. Introducing tables: rankings

Example 136: ★★ **Mutt's Adventure** Awarding points for visiting a room for the first time.

Example 137: ★★★ **No Place Like Home** Recording a whole table of scores for specific treasures.

§9.3. Introducing tables: rankings

Another tradition of interactive fiction is that the player has a current 'rank' according to how far his or her score has got. We can (but need not) choose to provide such rankings, and should do so by specifying a table like this:

```
Table 1 - Rankings

Score    Rank
0        "Beginner"
25       "Amateur Adventurer"
50       "Novice Adventurer"
100      "Junior Adventurer"
200      "Adventurer"
300      "Master"
330      "Wizard"
```

Typographically, tables in Inform look as much as possible like those found in non-fiction books: they can have many columns, so this is only a simple example (drawn from the actual rankings used by Infocom's *Zork I*, 1979). Each line in the source represents one row in the table, and the entries on a line must be separated by at least one tab character. (An entry might of course have several words with spaces in between, so a space is not enough to separate entries: this is the only context when Inform distinguishes between spaces and tabs.) The table must occupy a single whole paragraph, with no skipped lines or missing entries. We are free to use extra tabs to indent it if we like.

Ordinarily we must explicitly ask to use the information presented in a table, but the table of rankings is a special case: Inform uses it automatically, if it is provided at all. To be recognised it must have the name "Rankings" and must have two columns named and arranged as shown. The scores should be given in ascending order. Customarily, the score in the final row should be the maximum possible achieved in the story - so that only a player with maximum possible score can be awarded the final ranking - and the value of "maximum score" is automatically set to this bottom-row value if it has not been explicitly set anywhere else in the source text.

Start of Chapter 9: Time
Back to §9.2. Awarding points
Onward to §9.4. When play ends

§9.4. When play ends

Short of something like a power cut, the story can only end when one of the two participants chooses to end it: either the player, by deciding that enough is enough, or us.

In story-telling, there are many kinds of ending: happy, sad, decisive, bittersweet, surprise. Inform doesn't try to interfere in that kind of artistic choice, but it does need to know one thing about the ending: is it final, or not? Many authors like to make additional menu items available if the player has completed the story right through, but not if she has reached an early or wrong ending. (See the activity "amusing a victorious player", for example.)

end the story

This phrase ends the story at the next opportunity (typically as soon as the current rule ends), with the closing message "The End." The end is not

considered final.

end the story finally

This phrase ends the story at the next opportunity (typically as soon as the current rule ends), with the closing message "The End." The end is considered final, and any hidden menu options will be revealed.

end the story saying (text)

This phrase ends the story at the next opportunity (typically as soon as the current rule ends), with the closing message given in the text. The end is not considered final. Example:

```
end the story saying "You have been stymied"
```

end the story finally saying (text)

This phrase ends the story at the next opportunity (typically as soon as the current rule ends), with the closing message given in the text. The end is considered final, and any hidden menu options will be revealed. Example:

```
end the story finally saying "You have defeated Sauron"
```

The closing message is traditionally set out in asterisks:

```
*** The End ***
```

This style is traditional in IF, and goes back to 1980 if not earlier.

We can test the current state like so:

if story has ended:

This condition is true if an end has been declared using one of the "end the story..." phrases.

if story has not ended:

This condition is true if no end has been declared using one of the "end the story..." phrases.

if story has ended finally:

This condition is true if an end has been declared using one of the "end the story finally..." phrases, so that an ending has been reached which the author feels is a completion of the player's experience.

if story has not ended finally:

This condition is true if an end has been declared using one of the "end the story..." phrases, but not "finally", so the author feels that the player can get further experience by playing again and trying different approaches.

The rulebook "when play ends" is the matching bookend to "when play begins". It is followed when the story decides to end (not when the player simply gives up and quits), and before any epitaph like

```
*** You have been poisoned ***
```

appears. For example:

```
When play ends, say "Oh dear."
```

Surprisingly, the end is not always the end:

resume the story

This phrase causes an ended story to resume exactly as if no "end the story..." phrase had been used. Example:

```
When play ends:
    if the story has not ended finally:
        say "Oh dear. Still, here's another chance.";
        resume the story.
```

The phrase is likely to be sensible only as part of a "when play ends" rule. Other traditional uses include giving the player three lives, as in an old-school arcade machine.

Start of Chapter 9: Time

Back to §9.3. Introducing tables: rankings

Onward to §9.5. Every turn

Example 138: ★★★ **Big Sky Country** Allowing the player to continue play after a fatal accident, but penalizing him by scattering his possessions around the game map.

§9.5. Every turn

The passage of time in interactive fiction is broken up into a succession of turns, in each of which the player types a request and is given a response. Usually each such request triggers one action, but sometimes a whole sequence are fired off, as when the player types "get all" in a cluttered room.

As we've seen, the variable "turn count" holds the number of turns of play so far. By convention turn number 0 is the time when Inform prints up the banner and any initial text; it becomes turn number 1 when the player's first command is typed.

One of the last things to happen in each turn is that Inform will apply any rules which have been set to occur "every turn", like so:

```
Every turn, say "The summer breeze shakes the apple-blossom."
```

This is equivalent to writing:

```
An every turn rule: say "The summer breeze shakes the apple-
blossom."
```

Note that the text about blossom, which will quickly become tiresome, is said at the end of every turn, not at the beginning, and in particular not before the player's first opportunity to type a command.

As usual when defining rules, we can add stipulations: any condition can be attached using "when".

```
Every turn when the location is the Orchard, say "The summer
breeze shakes the apple-blossom."
```

```
Every turn when the player can see the rotting fish, say "Your nose
twitches involuntarily."
```

Start of Chapter 9: Time

Back to §9.4. When play ends

Onward to §9.6. The time of day

Example 139: ★★★ **Witnessed 1** A kind of battery which can be put into different devices, and which will lose power after extended use.

Example 140: ★★★★ **Text Foosball** A game of foosball which relies heavily on every-turn rules.

§9.6. The time of day

Inform keeps track of the time of day automatically: play ordinarily begins at 9 AM and each turn takes one minute. In some works of interactive fiction, time of day is crucial:

in others, it is irrelevant or even, by a sort of tacit convention, non-existent. So Inform does nothing to act upon the time, or to reveal it to the player, without instruction. Nevertheless it is there.

A sentence like the following allows the initial time to be set up as something other than 9 AM:

```
The time of day is 3:13 PM.
```

Here, "3:13 PM" is a constant value of a kind not seen before: it's a kind of value called "time", and the value "time of day" is a time that varies. After one turn it will be 3:14 PM, then 3:15 PM and so on.

Note that the sentence above is an assertion (a statement about the initial state of affairs), not an instruction which can be part of a rule. It would be equivalent to write:

```
When play begins: now the time of day is 3:13 PM.
```

We more often change "time of day" to take care of drastic events:

```
At the time when the player loses consciousness:
    now the time of day is 10:12 AM;
    say "A mist comes over your vision, and when you come to, it is
morning and you are in bed."
```

Start of Chapter 9: Time

Back to §9.5. Every turn

Onward to §9.7. Telling the time

Example 141: ★★ **IPA** Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out if he overstays his welcome.

§9.7. Telling the time

Now that we have the time of day, we can of course use this value in room descriptions and the like:

```
The Clock Chamber is a room. "The dark chamber behind the clock
face, a mill-room of gears which grind down the seconds. Through
the glass you can see the reversed hands reading [the time of day]."
```

It seems odd, though, to read a precise numerical description of the time here: after all, it isn't a digital clock. A friendlier version would use:

say "[(time) **in words]"**

This text substitution produces the given time written out in English sentence form. For example:

```
"Through the glass you can see the reversed hands reading
[the time of day in words]."
```

might produce

```
Through the glass you can see the reversed hands reading
twenty to nine.
```

To reiterate an example which came up earlier, we could even work the time of day into the command prompt, which would lend the proper sense of urgency to a story played out against the clock:

```
When play begins: now the command prompt is "[time of day] >".
```

Start of Chapter 9: Time

Back to §9.6. The time of day

Onward to §9.8. Approximate times, lengths of time

Example 142: ★ **Situation Room** Printing the time of day

in 24-hour time, as in military situations.

§9.8. Approximate times, lengths of time

Clocks and watches vary considerably in how much detail they show, and we tend not to report the time over-precisely: half-past ten is an elastic concept. The following room description for the Clock Chamber comes across much more naturally:

```
The Clock Chamber is a room. "The dark chamber behind the clock
face, a mill-room of gears which grind down the seconds. Through
the glass you can see the reversed hands reading [the time of day to
the nearest five minutes in words]."
```

The phrase "... to the nearest ..." rounds off the given time, just as it sounds; as we'll see later, it can actually round off any arithmetic values, not just times. For instance, "9:58 PM to the nearest ten minutes" is 10:00 PM.

In talking about lengths of time, rather than times of day, it's useful to have these:

(number) **minutes ... time**

This phrase converts numbers into lengths of time. Example:

```
15 minutes
```

Because it's a phrase, not just a notation for writing constants down, the number doesn't have to be given literally:

```
let X be 5;
if the player is in the Slow Room, now X is 10;
let deadline be the time of day plus X minutes;
```

Note that lengths of time can't exceed 1440 minutes.

(number) **hours ... time**

This phrase converts numbers into lengths of time. Example:

```
10 hours
```

Note that lengths of time can't exceed 24 hours.

Start of Chapter 9: Time
Back to §9.7. Telling the time
Onward to §9.9. Comparing and shifting times

§9.9. Comparing and shifting times

Carrying out easy calculations with times is straightforward:

```
The chronometer is in the Clock Chamber. "On one wall is a terribly self-important chronometer showing the time in major world cities. London: [time of day]. Paris: [one hour after the time of day]. Tokyo: [9 hours after the time of day]. Cupertino, California: [7 hours before the time of day]."
```

Here we are using two phrases:

(time) **before** (time) **... time**

This phrase produces a time earlier by the amount given, keeping within the 24 hour clock. Example:

```
7 hours before 5:30 AM
```

produces 10:30 PM.

(time) **after** (time) **... time**

This phrase produces a time later by the amount given, keeping within the 24 hour clock. Example:

```
9 hours after 11 AM
```

produces 8 PM.

Similarly, we have conditions:

if (time) **is before** (time)**:**

This condition is true if the first time occurs earlier in the day than the second. In recognition of the fact that very few stories begin before 4 AM, whereas many run on past midnight, the start of the day is taken to be 4 AM: thus 3:59 AM is after 11:10 PM, but 4:04 AM is before it.

if (time) **is after** (time)**:**

This condition is true if the first time occurs later in the day than the second. In recognition of the fact that very few stories begin before 4 AM, whereas many run on past midnight, the start of the day is taken to be 4 AM: thus 3:59 AM is after 11:10 PM, but 4:04 AM is before it.

Start of Chapter 9: Time

Back to §9.8. Approximate times, lengths of time

Onward to §9.10. Calculating times

§9.10. Calculating times

We will occasionally need to perform more complex calculations with time, and in order to do that, we have a way to convert the time of day to numbers. Thus the phrase "the minutes part of ..." takes a time and produces a number from 0 to 59; similarly "the hours part of ..." extracts a number from 0 to 23, using the twenty-four hour clock.

minutes part of (time) **... number**

This phrase converts a time to a number, then takes the result mod 60, which in effect produces the number of minutes after the hours are thrown away. Example:

```
minutes part of 12:41 PM
```

produces 41.

hours part of (time) **... number**

This phrase converts a time to a number, then divides the result by 60, which in effect produces the number of hours after minutes are thrown away. Example:

```
hours part of 8:21 AM
```

produces 8.

To go the other way, we can convert any number to a duration by writing "minutes" or "hours" after it. For instance:

```
The clock error is a number that varies. To thump the mechanism:
now the clock error is a random number from -10 to 10.
```

```
The broken grandfather clock is in the Chamber. "An erratic
grandfather clock seems to say it is [clock error minutes after the time
of day]."
```

```
When play begins, thump the mechanism. Instead of attacking the
broken clock: thump the mechanism; say "You thump the clock,
which now reads [clock error minutes after the time of day].".
```

Note that "clock error" is a number, but "clock error minutes" is a time.

Start of Chapter 9: Time

Back to §9.9. Comparing and shifting times

Onward to §9.11. Future events

§9.11. Future events

We often want to arrange for something to happen at some point in the future. Here is yet another timepiece:

```
An egg-timer is in the Chamber. "A plastic egg timer in the shape of a
chicken can be pressed to set it going."

Instead of pushing the egg-timer:
    say "It begins to mark time.";
    the egg-timer clucks in four turns from now.

At the time when the egg-timer clucks:
    say "Cluck! Cluck! Cluck! says the egg-timer."
```

The event here is called "the egg-timer clucks". It only happens if we instruct so, using one of the following phrases:

(rule) **in** (time) **from now**

This phrase causes the given rule to be run at a given time offset from the current time of day. Example:

```
    the egg-timer clucks in 18 minutes from now;
```

(rule) **in** (number) **turn**/turns **from now**

This phrase causes the given rule to be run at a given number of turns after the current one. Example:

```
the egg-timer clucks in four turns from now;
```

(rule) **at** (time)

This phrase causes the given rule to be run at a given time of day. Example:

```
the egg-timer clucks at 11:35 AM;
```

If we know in advance what time we want something to happen, we can more simply write:

```
At 4 PM: say "The great bells of the clock tower chime four."
```

(Note that in either case such rules begin with the word "at": they are the only rules allowed to begin with the word "at".)

A small warning: timed events like these only have a chance to occur during the turn sequence, that is, once every turn. In most stories, one turn takes one minute, so there will in due course be a turn happening at exactly (say) 11:35 AM. But if the clock is being advanced faster than this, it's possible that there are turns at (say) 11:32 AM and then not until 11:37 AM. But an event set for 11:35 AM will nevertheless happen -- it will run at the first available turn after that time, which will be 11:37 AM. Events can thus happen up to half an hour late, though Inform cancels them if the elapsed time is greater than that.

The Scenes panel of the Index can be a useful way to see what events have been set.

Start of Chapter 9: Time

Back to §9.10. Calculating times

Onward to §9.12. Actions as conditions

Example 143: ★ **MRE** Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

Example 144: ★★ **Totality** To schedule an eclipse of the sun, which involves a number of related events.

Example 145: ★★ **Empire** A train which follows a schedule, stopping at a number of different locations.

Example 146: ★★★ **Hour of the Wren** Allowing the player to make an appointment, which is then kept.

---

§9.12. Actions as conditions

There are two ways that descriptions of actions can be used as conditions. First, we can simply describe an action, and then the condition will be true if that is what the player is trying to do, and not otherwise:

```
if taking a container, ...
```

This is actually an abbreviation for the longer, some would say preferable form:

```
if we are taking a container, ...
```

Secondly, we can talk about the past as well as the present, which is very useful since interactive fiction often contains situations which are changed by earlier events.

```
Instead of waiting when we have taken the lantern, say "No, your
acquisitive nature is roused now, and simply waiting will no longer
do."
```

More on the past tense later follows in the next section: note that "we are taking" has become "we have taken". For the rule to apply, it is not enough that the action "taking the lantern" has been tried: it must have succeeded. Note also that it's enough for any actor in the story to have successfully taken the lantern: it doesn't have to be the player.

Start of Chapter 9: Time

Back to §9.11. Future events

Onward to §9.13. The past and perfect tenses

Example 147: ★ **Night Sky** A room which changes its description depending on whether an object has been examined.

Example 148: ★★★ **Zero** A box which called "horribly heavy box" after the player has tried to take it the first time.

§9.13. The past and perfect tenses

The remaining sections of this chapter go into more technical ways to think about the progress of the story through time, and can be skipped at a first reading.

Conditions are clauses which require Inform to make a decision: is such-and-such true, or not true? We have already seen conditions attached to rules using "when":

```
Instead of waiting when the Sorting Hat is in the Hall: ...
```

and, as we shall later see, we will often want to write instructions like:

```
if the Sorting Hat is in the Hall, say "Hermione blinks apprehensively."
```

The condition is "the Sorting Hat is in the Hall", and during play this will sometimes be true and sometimes false.

A condition in the form "X is Y" is of course written in the present tense, and refers to the current state of affairs. Three other tenses are allowed. First, the present perfect:

```
if X has been Y ...
```

is true if it has ever been the case that "X is Y" at the start of any turn (or any action). So, for instance,

```
if the gate has been open ...
```

will be valid if and only if the gate has ever been made open by any action (even if it is closed now), or if it started out by being open when play began.

Next is the past tense:

```
if X was Y ...
```

holds if and only if "X is Y" was true at the start of the most recent action. This is convenient when trying to describe what has changed in the course of the action, but sometimes also when making the action itself happen. For instance:

```
if the lantern was switched on, now the lantern is switched off;
if the lantern was switched off, now the lantern is switched on;
```

Completing the set is the past perfect:

```
if X had been Y ...
```

which records whether "X has been Y" was true at the start of the most recent action. All these verbs can of course be negated (though "wasn't" and "hadn't" are disallowed as poor style: we use "was not" and "had not" instead). So for example,

```
if the player had not been in the Ballroom ...
```

is true if the player hadn't visited the Ballroom at the start of the most recent action.

Something we must watch out for is that variables might not have the same values in the past that they have now. As a result, writing conditions such as "if the noun has been open" is a bad idea, because in the past "the noun" very likely referred to

something different. It is really only safe to talk in the past tense about definite, fixed things: "if the Great Gates of Kiev have been open" would be fine.

Start of Chapter 9: Time

Back to §9.12. Actions as conditions

Onward to §9.14. How many times?

Example 149: ★ **Tense Boxing** An overview of all the variations of past and present tenses, and how they might be used.

Example 150: ★★ **Bruneseau's Journey** A candle which reacts to lighting and blowing actions differently depending on whether it has already been lit once.

Example 151: ★★ **Elsie** A door that closes automatically one turn after the player opens it.

---

§9.14. How many times?

There are two further ways to examine the historical record. Given any condition, we can say

```
if (...condition...) for the second time ...
if (...condition...) twice ...
if (...condition...) 2 times...
if (...condition...) two times...
```

(all of which are synonymous: the words once, twice, thrice, one, two, three, four, five, six, seven, eight, nine, ten, first, second, third, fourth, fifth, sixth, seventh, eighth, ninth and tenth all mean what they obviously should). The result is true if the condition holds now and has held for only one previous spell in the past. A condition holding for, say, fifteen consecutive turns without a break counts as only one "time" - so what we mean by "twice" here is that it is true now, was previously false for a while, and was previously true for a while before that, but no more. In effect, then,

if the player is in the Ballroom for the third time ...

is true if this is the third visit to the Ballroom. We can also say

if the player is in the Ballroom for more than the third time ...

or similarly "less than", "at least", "at most". It would be more natural, though, to say

if the player has been in the Ballroom three times ...

The adjective "only" (or equivalently "exactly") can be added to obtain

if the player has been in the Ballroom only three times ...

To recap, this means there have been exactly three visits to the Ballroom in history, whereas

if the player is in the Ballroom for the third time ...

means there have been exactly three visits, the third of which is still going on - an important distinction.

Start of Chapter 9: Time

Back to §9.13. The past and perfect tenses

Onward to §9.15. How many turns?

Example 152: **Infiltration** A room whose description changes depending on the number of times the player has visited.

---

§9.15. How many turns?

So much for "times" - spells in which a condition is true. We can also test the length of time, in turns of play, that something has been true. Thus:

```
if ... for three turns;
```

means that the condition holds now, and held at the start of this turn, at the start of last turn, and at the start of the turn before that. In particular:

```
if the floppy hat has been worn for three turns ...
```

will be false if the hat is not currently worn (even if it has been often in the past) and, on the other hand, will be true if the hat has been worn for twenty turns. Here again we can be more specific. These are synonymous:

```
if the floppy hat is worn for the third turn ...
if the floppy hat has been worn for only 3 turns ...
if the floppy hat has been worn for exactly three turns ...
```

all requiring that the hat wasn't worn four turns ago. As before, "more than", "less than", "at least" and "at most" so forth can also be used - say, "for at least 21 turns".

A warning: we must be careful when writing something like

```
if the noun has been open ...
```

since this tests whether it has ever been true that the noun of the then action was open: not whether the current noun-object has ever been open.

Lastly, note that the beginning of play - when (usually) initial text and a banner is printed, followed by a room description - counts towards these counts. In effect, this is a turn: one in which the player compulsorily performs the looking action, rather than being asked for a command. (By convention it is numbered as turn number 0, and doesn't contribute towards the turn count.)

Start of Chapter 9: Time

Back to §9.14. How many times?

Onward to Chapter 10: Scenes: §10.1. Introduction to scenes

Example 153: **Annoyotron Jr** A child who after a certain period in the car starts asking annoying questions.

Examples from Chapter 9: Time

Start of this chapter
Chapter 10: Scenes
Indexes of the examples

135 **ExampleClueless** *RB*

A murderer for the mystery is selected randomly at the beginning of the game.

"When play begins" is the best point to initialize any aspects of the game that are meant to change between playings. For instance, in this scenario, we would randomly select one of the other characters to be guilty of murder:

```
"Clueless"

The murderer is a person that varies.

When play begins:
    now the murderer is a random person who is not the
player.

The Billiards Room is a room. Colonel Mustard and
Professor Plum are men in the Billiards Room. Miss
Scarlet and Mrs White are women in the Billiards Room.
```

```
Instead of examining the murderer:
    say "[The noun] certainly looks fiendish!"


Test me with "x mustard / x plum / x scarlet / x white".
```

Example **Mutt's Adventure** *RB*

Awarding points for visiting a room for the first time.

Suppose we want to reward the player the first time he reaches a given room. The "unvisited" attribute is useful for this: unlike such constructions as "going to a room for the first time", it doesn't develop false positives when the player has merely tried to go to the room in question. "Every turn when the player is in a room for the first time" is also unhelpful, because it continues to be true as long as the player is in a room on his first visit there.

```
"Mutt's Adventure"


Use scoring.


Section 1 - Procedure


A room can be scored or unscored.


Carry out going to a unvisited scored room:
    increment the score.


Section 2 - Scenario
```

The Incan Palace Compound is a room. "After numerous false leads through the jungles of Peru, and an arduous trek along the Amazon, you have arrived, at last, here: at Atagon, the lost city of untold treasure."

The startlingly intricate door is a door. It is inside from Incan Palace Compound and outside from the Treasure Room. "A door carved all over with figures of ancient gods, and protected by an assortment of gears and latches, [if open]stands open[otherwise]blocks progress[end if] towards [the other side of the intricate door]."

The description of the Treasure Room is "To your considerable surprise, the treasure room is stocked with art objects from a vast range of eras and geographical locations: beside the expected pre-Columbian gold there are Cycladic figurines, Chinese Tang-dynasty pottery, purses that might have been stolen from Sutton Hoo. [one of]If the British Museum developed a nasty expectorant cough, this is what you'd find in its hanky. [or][stopping]".

The Treasure Room is scored.

Test me with "in / out / in".

Example **No Place Like Home** *RB*

Recording a whole table of scores for specific treasures.

Suppose we want to assign scores for a whole range of objects the player might pick up. One systematic way to do this would be with a table of point values for things:

"No Place Like Home"

Use scoring.

The Hall of the Gnome King is a room. The emerald cow is a thing in the Hall of the Gnome King. The ivory chessman is a thing in the Hall of the Gnome King. The book of incantations is a thing in the Hall of the Gnome King.

Table of Point Values

item	score
cow	10
incantations	4
chessman	1

Report taking an item listed in the Table of Point Values:
    increase the score by the score entry;
    blank out the whole row.

Test me with "take all".

"Blank out the whole row" removes the line from the table, so that each award will occur only once. The player will not be able to earn more and more points by dropping and taking the same item again.

ExampleBig Sky Country *RB*

Allowing the player to continue play after a fatal accident, but penalizing him by scattering his possessions around the game map.

Some older games allowed the player to be resurrected after a death, but punished him by distributing his possessions far and wide. Here we emulate that effect.

"Big Sky Country"

Use scoring.

When play begins: say "There's a bit of a drive over from Anaconda, Montana, and then through a couple or three ghost towns, but finally you find what you're looking for, and strike out on foot..."

Entrance to Devil's Canyon is a room. "You are at the top of a steep road, which proceeds down into the canyon proper." A sign is in Devil's Canyon. It is fixed in place. "An ominous sign has been put up by the local sheriff's office." The description is "PROCEED AT OWN RISK - NO RESCUES!"

Instead of going down when a random chance of 1 in 3 succeeds:
    say "Whoooops, your footing is not as secure as you thought...";
    end the story.

Dusty Path is below Entrance. "A dusty path, with grey-brown thorny bushes on either side. Immediately to your right is a sheer drop; far below you can see the rusting remains of a Model T that some fool tried to drive by here."

Hairpin is below Dusty Path. "A sharp bend in the road, doubling back down towards the bottom of the canyon.

```
Just north of here there is also a small cavern of some
kind[if the stick pin is in the cavern], which attracts your
eye with some glittery thing[end if]."
```

```
The Cavern is north of Hairpin. "Really not much more
than a little hollow in the side of the canyon." In the
cavern are a snake and a diamond stick pin. The snake
is an animal. The description of the snake is "You're no
expert, but it looks like a rattler."
```

```
Instead of taking the diamond stick pin in the presence
of the snake: say "Turns out the snake is partial to that
there pin, and takes exception to your intending to make
off with it."; end the story.
```

In a fuller implementation of this game, we might make it possible to get by the snake, but in this version, it's just going to remain troublesome.

```
Crooked Path is below Hairpin. "You're about two thirds
of the way down to the bottom of the cavern at this
point."
```

```
At the Spot is below Crooked Path. "This'll be it: a bare
patch of ground that might as well have an X painted
right on it."
```

```
Rule for supplying a missing noun while digging:
    now noun is the location.
```

```
Understand "dig" or "dig hole/here" or "dig in ground/dirt/
earth" as digging. Digging is an action applying to one
thing.
```

```
Instead of digging at the spot:
```

say "You dig and dig, and after a half hour or so, sure enough, you do turn up a big box of gold! You're going to be richer than God and Bill Gates put together.";
increase the score by 5;
end the story finally.

Instead of digging at the spot when the player does not carry the shovel:
say "What, without your shovel? That won't work too well."

The player carries a walking stick. The player wears a hat, a whistle, and a daypack. The daypack contains a mylar blanket, a granola bar, a cellular phone, a water bottle, a folding shovel, and a photocopied map. The granola bar is edible. Instead of drinking the water, say "You quench your thirst, for the time being." The description of the map is "The map shows the winding path of Devil's Canyon, with a large X down by the south end. That would be where your uncle Jesse buried the gold from the train robbery."

The maximum score is 5.

When play ends when the story has not ended finally:
say "Oh dear, that ought to be fatal! However, if you like I can get you out of it...

Shall I? >";
if the player consents:
repeat with item running through things had by the player:
move the item to a random visited room;
say "A strong wind picks you up and sets you back at [the location], though perhaps minus a few of your things.";
resume the story;
try looking.

"If the player consents" is just a convenient way to ask a yes/no question that the player must answer before going on with the game.

Example **Witnessed 1** *RB*

A kind of battery which can be put into different devices, and which will lose power after extended use.

The following example makes fairly ample use of material that we haven't seen yet, but gives some idea of the flexibility of every turn rules. Suppose we want to have a number of electrical devices, all of which may be powered by a set of batteries. The batteries will all need to be discharged as they are used (regardless of what device they happen to be controlling at the moment). So:

```
"Witnessed"

A battery is a kind of thing. A battery has a number
called charge. The charge of a battery is usually 15.

Every turn:
   repeat with hollow running through battery
compartments:
      if the hollow is part of a switched on device (called
the machine):
         if a battery (called cell) is in the hollow:
            decrement the charge of the cell;
            carry out the warning about failure activity with
the machine;
            if the cell is discharged, carry out the putting
out activity with the machine;
         otherwise:
            carry out the putting out activity with the
machine.

Warning about failure of something is an activity.
```

```
Rule for warning about failure of a device (called the
machine):
    if a random battery compartment which is part of the
machine contains a battery (called the power source):
        if the charge of the power source is 2, say "[The
machine] is obviously going to go out quite soon."



Putting out something is an activity.



Rule for putting out a device (called the machine):
        say "[The machine] loses power and switches off!
[line break]";
        silently try switching off the machine.



A battery compartment is a kind of container. A battery
compartment is usually closed and openable. One
battery compartment is part of every device. Instead of
inserting something which is not a battery into a battery
compartment, say "Only batteries should go in a battery
compartment."
```

And to get rid of annoying messages like "Which would you like to close, the flashlight or the flashlight's battery compartment?" when only the compartment is closable, we might add some understanding instructions:

```
Understand "turn on [device]" as switching on.



Understand "turn off [device]" as switching off.



Understand "open [openable closed thing]" as opening.



Understand "close [openable open thing]" as closing.
```

Understand "put [something] in [container]" as inserting it into.

Instead of opening a device, try opening a random battery compartment which is part of the noun. Instead of closing a device, try closing a random battery compartment which is part of the noun. Instead of inserting a battery into a device, try inserting the noun into a random battery compartment which is part of the second noun.

Instead of switching on an empty device:
    say "Nothing happens, perhaps because there isn't a charged battery in [the noun]."

Instead of switching on a battery compartment which is part of a device (called the power user), try switching on the power user.

Definition: a device is empty:
    if a random battery compartment which is part of it contains a battery (called the power source):
        if the power source is discharged, yes;
        no;
    yes.

Definition: a battery is discharged if its charge < 1.

A light source is a kind of device. Carry out switching on a light source: now the noun is lit. Carry out switching off a light source: now the noun is unlit.

The flashlight is a light source. A D battery is a battery carried by the player.

The cassette recorder is a device. Every turn: if the cassette recorder is switched on, say "The cassette recorder hisses faintly."

Rule for warning about failure of the cassette recorder:
    if a random battery compartment which is part of the cassette recorder contains a battery (called the power source):
        if the charge of the power source is 2, say "The hiss from [the cassette recorder] begins to warble ominously."

The player wears a backpack. The backpack is openable. In the backpack is the flashlight and the cassette recorder.

The description of the cassette recorder is "Useful both for recording your notes and for capturing any odd ghostly sounds you may hear."

The description of the backpack is "An old familiar pack, which you know so well that you can find all its pockets and take things in and out of it in pitch darkness. To avoid it showing up oddly in photographs, it is entirely black, with no shiny or metallic attachments."

The description of the flashlight is "You bought a new one just for this occasion, because you were worried about bringing something too small or light. This is a heavy-duty flashlight with an adjustable-focus beam. The case is made of metal, rather than plastic, and there is a spare light-bulb inside as well. You've put a band of masking tape around the handle and written in

your initials in red marker.

There is a piece of red cellophane attached to the business end of the flashlight to keep it from being overly bright."

The red cellophane is part of the flashlight.

Instead of doing something to the red cellophane: say "You need the cellophane on the flashlight so that using it does not completely destroy your night vision."

Thirtieth Street Station is a room. "A huge, high, rectangular room with coffered ceilings, which looks grand but mostly makes you feel lonely and small. There are long benches in rows down the middle of the room, and an information desk with the train times, and a series of ticket windows, none of which matters very much at the moment."

The benches are an enterable supporter. They are scenery in the Station. The information desk is scenery in the Station. Some ticket windows are scenery in the Station. Instead of examining scenery in the Station: say "You're fairly sure that whatever is going on here has nothing to do with [the noun]." Understand "window" as ticket windows.

The mural is fixed in place in Thirtieth Street. "At the north side of the station is a particularly pointless and empty annex to the main room. It is dominated by a huge relief of sorts, and this is what you remember." Understand "metal" or "relief" or "huge" as the mural. The description of the mural is "It is both stylized and confusing, but you think it might be supposed to represent the various tasks and occupations of

```
Philadelphia's population. The portions closer to the
ground look as though they have recently been
subjected to a light dusting of talcum powder. No
unusual prints are evident."
```

```
The wind chimes are fixed in place in Thirtieth Street.
"Carefully attached to the wall with a piece of duct tape
and a hook is a light-weight set of wind chimes.
Someone else has been here before you, it seems."
The description is "Several of your friends use wind
chimes as a sort of ghost alarm, since ghosts
sometimes cause very localized movements of air when
there is no natural breeze."
```

And this last bit, borrowed from the chapter on Understanding, adds some special instructions to help Inform decide when the player is likely to be referring to a compartment and when he's likely to be referring to the device itself.

```
Does the player mean doing something other than
searching to a battery compartment: it is unlikely. [We
discourage Inform from choosing a compartment when
the player uses just the name of a device or the word
'battery'.]
```

We also need to deal with commands like PUT BATTERY IN FLASHLIGHT, where Inform might construe BATTERY as the D battery, the flashlight's battery compartment, or the cassette recorder's battery compartment -- and might also construe FLASHLIGHT as either the flashlight's battery compartment or the flashlight itself.

```
Does the player mean inserting into a battery
compartment:
    if the noun is nothing:
        it is very likely;
    otherwise:
        make no decision.
```

```
Does the player mean inserting a battery compartment
into: it is very unlikely.
```

Does the player mean inserting something into a device: it is unlikely.

Does the player mean searching a battery compartment: it is very likely.

Test me with "test first / test second".

Test first with "i / open flashlight compartment / put battery in it / turn on flashlight / take d battery / open cassette compartment / turn on cassette / put battery in cassette compartment / turn on cassette / z / z / z / z".

Test second with "get d battery / put d battery in flashlight compartment / turn on flashlight / z / z / z / z / z / z / turn off flashlight / z / z / turn on flashlight / z".

---

140 **Example** **Text Foosball**
A game of foosball which relies heavily on every-turn rules.

RB

Suppose we want a game of foosball in which our opponent acts every turn, but does different things depending on where the ball currently lies. We can put together a sequence of every-turn rules to account for this, as follows:

"Text Foosball"

Use scoring.

The Lounge is a room. "The Lounge is appointed with everything necessary to rest and relaxation: a vending machine, a potted palm, a stack of Entertainment Weekly issues from 1993, and -- your pride and joy -- a foosball game."

The foosball game is scenery in the Lounge. Understand "table" or "football" or "foozball" or "fussball" or "soccer" as the foosball game. The game is a supporter. On the game is a small white ball. The ball can be still, approaching, receding, or unreachable. The description of the ball is "Currently [small white ball condition]."

After printing the name of the small white ball, say " ([small white ball condition])".

When play begins:
    now left hand status line is "You: [score]";
    now right hand status line is "Joey: [Joey's score]".

Some tiny men on sticks are part of the game. Understand "handles" as the tiny men. The description is "Okay, a couple of the tiny men have had their feet broken off, and the table surface itself is a bit warped, and the ball resembles a quail egg in respect of shape and color. This makes for a game of unusual randomness, but skill is overrated."

Instead of attacking or pulling or pushing the game when the ball is unreachable:
    say "You give the table a good shove, and the ball moves ever-so-slightly.";
    now the ball is still.

```
Instead of taking the white ball:
    say "You'd forfeit the game if you did that."


Instead of turning the tiny men when the ball is
unreachable:
    say "The ball has somehow gotten to a mystical point
on the table where it cannot be reached, no matter
what. Close inspection reveals that this point has been
marked in chalk with a tiny X. Not that that does any
good."


Instead of turning the tiny men when the ball is
approaching:
    if a random chance of 2 in 3 succeeds:
        if a random chance of 1 in 2 succeeds, now the ball
is receding;
        otherwise now the ball is still;
        say "[if the ball is still]Thunk. [otherwise]Thwack!
[end if]You keep the ball from reaching its goal! Now it is
[small white ball condition].";
    otherwise:
        let Joey score.


To let Joey score:
    now the ball is still;
    now Joey's score is Joey's score + 1;
    say "The ball rolls neatly into your goal, despite your
efforts. ";
    if Joey's score < score, say "You put the ball back in
the center with a snap. No reason to worry yet; you're
still ahead. Joey looks determined, though.";
    otherwise say "After allowing a moment or two for
Joey's gloating to pass, you replace it at the center."


Instead of turning the tiny men when a random chance
of 1 in 13 succeeds:
    if the ball is unreachable, continue the action;
    now the ball is unreachable;
    say "You hit the ball off-center and it rolls sluggishly
into a little dip in the surface of the foosball table. ";
    if Joey's score > 7, say "'You did that on purpose!'
```

```
Joey exclaims indignantly.";
    otherwise say "You and Joey exchange glances. This
is never good."



Instead of turning the tiny men:
    say "You madly rotate the tiny men on sticks! ";
    if a random chance of 1 in 2 succeeds:
        say "Hoorah! You hit the ball!";
        now the ball is receding;
    otherwise:
        say "Somehow you fail to bring your monopodal
player into contact with the ball."



Joey is a man in the Lounge. "Joey is hunkered over the
foosball handles on his side of the table." Joey can be
active or inactive.



Joey's score is a number that varies.



Every turn: now Joey is active.



Every turn when the ball is approaching and Joey is
active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    now Joey is inactive;
    let Joey score;
    rule succeeds.



Every turn when the ball is unreachable and Joey is
active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    now Joey is inactive;
    say "Joey glares angrily at the stuck ball."
```

```
Every turn when the ball is receding and Joey is active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    if the ball is unreachable, make no decision;
    now Joey is inactive;
    if a random chance of 1 in 2 succeeds:
        if a random chance of 1 in 2 succeeds, now the ball
is still;
        otherwise now the ball is approaching;
        say "Joey connects with your shot. Now the ball is
[small white ball condition]!";
    otherwise:
        now the ball is still;
        say "Joey tries to block, but misses! Back it goes in
the center, where it is [small white ball condition].";
        increment the score.


Every turn when the ball is still and Joey is active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    if the ball is unreachable, make no decision;
    now Joey is inactive;
    if a random chance of 1 in 2 succeeds:
        now the ball is approaching;
        say "Joey hits the ball solidly down towards your
goal. Now it is [small white ball condition].";
    otherwise:
        say "Joey fails to hit the ball in your direction. It
remains [small white ball condition]."


Every turn:
    let total be Joey's score + score;
    if total > 9:
        if Joey's score > score, end the story saying "Rats!
Joey wins!";
        if Joey's score < score, end the story finally saying
"Victory is yours!";
        if Joey's score is score, end the story saying "A
perfect tie."
```

Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out if he overstays his welcome.

Suppose we wanted a game set in a living town, with locations opening and closing at different times of day, and business carrying on as usual. The point might be to force the player to plan his itinerary carefully to hit the right spots at the right times; or we might be writing a more contemplative piece, where part of the enjoyment came from just watching the characters wander around doing their daily business...

```
"IPA"


When play begins: now the right hand status line is
"[time of day]".


The time of day is 9:50 AM.


A shop is a kind of room. A shop has a time called the
opening hour. The opening hour of the shop is usually 8
AM. A shop has a time called the closing hour. A shop
usually has closing hour 6 PM.


Check going to a shop (called the target):
    if the time of day is before the opening hour of the
target,
        say "[The target] is locked up until [the opening
hour of the target]." instead.


Check going to a shop (called the target):
    if the time of day is after the closing hour of the target,
        say "[The target] has been closed since [the
closing hour of the target]." instead.
```

Every turn when the location is a shop:
    let deadline be the closing hour of the location;
    if the deadline is before the time of day:
        let target be a random adjacent room which is not a shop;
        say "You are gently but firmly ushered out, since [the location] is closing.";
        move the player to the target.

The Strip-mall Parking Lot is a room. "Dead Christmas trees are heaped outside the bagel shop. Strips of dirty ice survive along the curb, and in the shadows of the lamp-posts. A wet, almost illegible sheet of algebra homework is plastered to the asphalt.

Pinewood Brewing Supply is at the east end of the lot."

Pinewood Brewing Supply is a shop. It is east of Parking Lot. The opening hour of Pinewood Brewing Supply is 10:00 AM. The closing hour of Pinewood Brewing Supply is 3:30 PM. "Shelves and shelves of malt and hops; large glass carboys, and plastic tubing; empty bottles; bottle-caps; bottle-labeling kits; starters for vinegar, sourdough, root beer.

A sweet malty smell hangs in the air."

Instead of going to Brewing Supply when the time of day is before the opening hour of Brewing Supply for the second time:
    say "You rattle at the door again. 'Hold your horses, for crying out loud,' yells a voice from within."

Noah's Bagels is a shop. It is north of the Parking Lot. The opening hour of Noah's Bagels is 6:00 AM. The closing hour of Noah's Bagels is 11:00 AM. "The

selection has been somewhat picked over, leaving you with your choice of Pumpernickel, Asiago, or Everything."

Test me with "e / e / n / z / s / e / z / e / z / z / e".

---

142 **Example** **Situation Room** *RB*

Printing the time of day in 24-hour time, as in military situations.

Though Inform normally prints times in AM/PM terms, it stores the hours and minutes as 24-hour time; so, if we like, we can easily extract that information again thus:

```
"Situation Room"

The Situation Room is a room.

To say (relevant time - a time) as 24h time:
    let H be the hours part of relevant time;
    let M be the minutes part of relevant time;
    say "[if H is less than 10]0[end if][H][if M is less than 10]0[end if][M]".

When play begins:
    now the time of day is 6:09 PM;
    now the right hand status line is "[time of day as 24h time]".

Test me with "z".
```

Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

Many older interactive fiction games required the player to find food and eat on a regular basis in order to avoid death. This effect was often unrealistic (since most people can survive much longer than a few hours without eating) and is often seen as an annoyance. However, for the sake of argument, suppose that we do want to construct a hunger-and-death system.

To make things a little more interesting, we will postulate that different foods are differently filling, so that if the player manages to find something really caloric, he is off the hook on his hunger search for a while.

We will also implement the system so that the player gets messages when he is hungry, then dies a short time later. (The times involved are ludicrously short, but this allows us to see the effects within a simple example. In a real game we would want to allow a considerably longer timer for the hunger to play out.)

First, a little scene-setting:

```
"MRE"


When play begins:
    now the right hand status line is "[time of day]";
    say "The procedure was painless at first: increased
strength was the first sign, followed by a sensation of
delayed time, as though everyone around you moved
more slowly. Your ability to dodge and perform feats of
agility doubled, then trebled. You were heralded as a
triumph of medicine. They told you there would be no
side effects worth speaking of.


They were wrong."
```

```
The Base Camp Larder is a room. "This room has been
reinforced after each incident -- and there have been
dozens in the last two months -- so that it now rivals Fort
Knox. Only your new skill and speed enabled you to
dodge the motion sensors, knock out the computerized
security system, fool the retinal scanner, and punch a
hole in the steel containment grating. But you're inside
now."
```

Now we define our food, and add some special instructions for what happens to our hunger counters when the food is eaten:

```
Food is a kind of thing. Food is usually edible. Food has
a time called the satisfaction period. The satisfaction
period of a food is usually 5 minutes.


A person can be hungry or replete. The player is
hungry.


The Larder contains an apple, a candy bar, and a large
plate of pasta. The apple, the candy bar, and the pasta
are food. The satisfaction period of the apple is 2
minutes. The satisfaction period of the pasta is 125
minutes.


Check eating something which is not food:
    say "[The noun] might be edible, but it's not what
you'd consider really food."


Check eating something when the player is not hungry:
    say "You're not hungry right now."


Carry out eating something:
    now the player is replete;
    hunger resumes in the satisfaction period of the noun
from now.
```

The first of those two phrases, "now the player is replete", causes the player to cease to be hungry; the second one sets up a future event in which the hunger sets in again. The length of time until that event depends on how satisfying the specific food is. Now we define that event:

```
At the time when hunger resumes:
    starvation occurs in three minutes from now;
    now the player is hungry.
```

```
At the time when starvation occurs:
    if the player is hungry, end the story saying "You
have starved".
```

Note "if the player is hungry": it is possible that the starvation event will be set up but the player will eat before it occurs; in that case, we want it not to take effect.

And now, since we really ought to give the player some warning of what is happening to him:

```
Every turn when the player is hungry:
    choose a random row in the Table of Hunger
Complaints;
    say "[hunger entry][paragraph break]".
```

```
Table of Hunger Complaints
hunger
"Gosh, you're starving."
"It feels as though you haven't eaten in days. Weeks,
almost."
"The world seems to slow down and everything
becomes sharper and brighter. You are a hunter, a
hunter of foodstuffs."
"You find yourself staring at [the random visible thing
that is not the player] and wondering how it would
taste."
```

```
Test me with "eat apple / z / z / z / eat candy bar / z / z /
z / z / z / z / z / z / z".
```

ExampleTtotality *RB*

To schedule an eclipse of the sun, which involves a number of related events.

First we define the events, and then we create a phrase to schedule them:

```
"Totality"

At the time when the penumbra appears:
    say "The sunlight dies away to an eerie, brownish
penumbra."

At the time when the eclipse begins:
    say "The moon passes across the disc of the sun,
plunging the world into darkness.";
    now the Chamber is dark.

At the time when the eclipse ends:
    say "The moon passes away from the sun, and a
gloomy penumbral light returns.";
    now the Chamber is lighted.

At the time when normal sunlight returns:
    say "The sun shines once more bright in the sky, not
to be eclipsed again on this spot for another thirty-seven
years."

To schedule an eclipse for (totality - a time):
    the penumbra appears at two minutes before totality;
    the eclipse begins at totality;
    the eclipse ends at three minutes after totality;
    normal sunlight returns at five minutes after totality.
```

Now we make use of the new phrase:

```
When play begins, schedule an eclipse for 3:27 PM.

The Chamber is a room.

The time of day is 3:25 PM.

Test me with "z / z / z / z / z / z / z / z / z".
```

We shall see much more about creating phrases later. Their advantage is that they enable a complicated sequence of operations to be given a meaningful name, and that they can be re-used many times as needed.

ExampleEmpire

RB

A train which follows a schedule, stopping at a number of different locations.

Suppose we want to have a train which, at fixed times, arrives at and leaves stations. It should be possible for the player to get on and off the train when it is stopped, but not while the train is in motion.

```
"Empire"

The Empire Builder Train is a room. The Train has a
room called the station. The station of the Train is
Seattle.

The description of the Empire Builder is "One of the
(relatively) plush long-distance Amtrak trains. You're in
```

```
a two-story car with toilets and a cafe at one end, not
having sprung for a sleeper.


[if the station of the Train is the train]Outside the window
there is rapidly-passing countryside.[otherwise]Through
the windows you can see the [station of the Train] train
station.[end if]"


Instead of exiting when the player is in the Train:
    if the station of the Train is the Train:
        say "The train is not stopped at a station." instead;
    otherwise:
        move the player to the station of the train instead.


Before going outside when the player is in the Train:
    try exiting instead.


Before going inside when the player is in the station of
the Train:
    move the player to the Train instead.


Seattle, Edmonds, Everett, Wenatchee, and Spokane
are rooms. The description of a room is usually "The
scenic train station of [the location][if the location is the
station of the train].


The pompously-titled Empire Builder train is pulled up
here, soon to continue its journey towards Chicago[end
if]."
```

And now our schedule for the train -- somewhat truncated, admittedly, since the full three-day journey from Seattle to Chicago is a bit long even for an ambitious example.

At 4:45 PM:
    if the player is in the train or the player is in the station of the train, say "The train pulls out of [the station of the Train]!";
    now the station of the Train is the Train.

At 5:10 PM:
    now the station of the Train is Edmonds;
    if the player is in the train or the player is in the station of the train, say "The train pulls into Edmonds and comes to a stop."

At 5:17 PM:
    if the player is in the train or the player is in the station of the train, say "The train pulls out of [the station of the Train], running north along the shore towards Everett.";
    now the station of the Train is the Train.

At 5:39 PM:
    now the station of the Train is Everett;
    if the player is in the train or the player is in the station of the train, say "The train arrives in scenic Everett, WA: the last stop before it turns east and heads over the mountains."

At 5:44 PM:
    if the player is in the train or the player is in the station of the train, say "The train pulls out of [the station of the Train] and turns east.";
    now the station of the Train is the Train.

At 8:39 PM:
    if the player is in the train or the player is in the station of the train, say "In darkness the train rolls into Wenatchee; which is just fine, considering that there is nothing to see here at all.";
    now the station of the Train is Wenatchee.

```
At 8:44 PM:
    if the player is in the train or the player is in the
station of the train, say "The train pulls out of [the station
of the Train] and continues east through the darkness
towards Spokane.";
    now the station of the Train is the Train.
```

Playing this out would of course require near inhuman patience. Let's set things up so that the player at least doesn't have to wait too long for his first departure:

```
The time of day is 4:43 PM.
```

...and provide fair warning of how slowly time is elapsing.

```
When play begins:
    now the right hand status line is "[time of day]".
```

```
Test me with "out / in / z/ z / z / out / z / z / z / z / z / z
/ z / z / z / z / z / z / z / z / z / z / z / z / z / z / z / out / in
/ out / z / z".
```

```
Test more with "out / z/ z / z / z / z / z / z / z / z / z / z / z /
z / z / z / z / z / z / z / z / z / z / z / z / z / z / z / z".
```

ExampleHour of the Wren *RB*

Allowing the player to make an appointment, which is then kept.

Here we allow the player to set the time at which some future event is going to happen, rather than letting the game decide. We'll need to borrow the syntax for defining new actions from a later chapter:

"Hour of the Wren"

When play begins:
    say "You more or less stumble across them in Central Park: a disparate group of people, all of different ages, sitting in a circle. They aren't talking to one another -- in fact, they seem to be trying very hard to ignore one another, like people in the waiting room of an especially embarrassing kind of doctor. You are about to go around when a woman in a grey pressed suit comes up to you. Her suit-skirt is trimmed in lavender cord, and she looks as though she might have been extremely sharp-dressed in 1944. She hands you a card."

The Circle in the Grass is a room. "No one is looking at you, except for the secretary, if that is what she is."

The player carries a card. The description of the card is "Typed: 'Active astrology - dislike your fortunes? change your stars! - make an appointment now - hour of the wren STILL AVAILABLE.'".

The time of day is 1:55 PM.

Understand "pick [time]" or "choose [time]" or "make appointment for [time]" or "make an appointment for [time]" as making an appointment for. Making an appointment for is an action applying to one time.

Carry out making an appointment for:
    say "Fate cannot be commanded more than once."

Instead of making an appointment for the time understood for the first time:

```
    say "You settle on [the time understood] for your
appointment. The woman makes a note of it in an
appointment book, which she carries in a brown paper
bag. 'Excellent choice, ma'am,' she says in a low, urgent
voice. 'You'll be very satisfied.'";
    stars shift at the time understood.

Understand "hour of the wren" as 2:00 PM.

At the time when stars shift:
    end the story saying "insert cataclysm here".

Test me with "x card / make appointment for hour of the
wren / z / z / z / z".
```

---

147 Example Night Sky *RB*

A room which changes its description depending on whether an object has been examined.

Sometimes a nice effect is to change the way things are described depending on the information the player has gained in the course of play. We could for instance write this:

```
"Night Sky"

The Planetarium is a room. "[if we have examined the
sinister message]A dark room where it seems
something is about to jump out at you![otherwise]A
tranquil dark room with a ceilingful of stars.[end if]"

The sinister message is a thing in the Planetarium. "A
message is taped to the wall." The description is
```

"'BEWARE.'"

Test me with "look / x message / look".

On the other hand, beware that this would not work as desired:

"Night Sky"

The Planetarium is a room. "[if we have listened to the sinister message]A dark room where it seems something is about to jump out at you![otherwise]A tranquil dark room with a ceilingful of stars.[end if]"

The sinister message is a thing in the Planetarium. "A message plays very softly, so that you would have to listen to hear it." Instead of doing anything other than listening to the message: say "It's only a sound, after all.". Instead of listening to the sinister message: say "A voice whispers, 'BEWARE'."

Test me with "listen to message / look".

The reason is that our Instead rule has pre-empted normal listening, so Inform considers that we have never successfully heard the message. The moral here is that "if we have..." is useful for tracking events that otherwise proceeded completely normally (picking up ordinary objects, examining things); if we have used instead to make some change, we will have to use a different approach to record that the event did occur as scheduled.

ExampleZero

A box which called "horribly heavy box" after the player has tried to take it the first time. *RB*

The names of objects might want to change also:

```
"Zero"

Weight is a kind of value. The weights are light,
manageable, and horribly heavy. Everything has a
weight.

A thing is usually manageable.
```

This does require a rule for an activity (see the chapter on Activities), but it's a fairly straightforward one:

```
Before printing the name of a horribly heavy thing
(called weighty object):
    if we have taken the weighty object, say "[weight] ".

The Nearly Empty Living Room is a room. "Nearly
everything is out, all of Helen's possessions and most of
yours." A man called Mr Zero is in the Nearly Empty
Living Room. "Mr Zero, despite being heavily paid to
assist in this operation, is giving you a look that clearly
conveys his lack of interest in budging even one more
item." The description of Mr Zero is "Many muscles, no
hair."

The book box is a horribly heavy thing in the Living
Room. The clothing box is a manageable thing in the
Living Room. A broom, a dustpan, some packing tape,
and a discarded newspaper are light things in the Living
Room.

After taking a horribly heavy thing:
    say "Taken. (Oof.)"
```

Test me with "get clothing / look / get book box / look / i / drop book box / look".

---

149 **Example** **Tense Boxing** *RB*

An overview of all the variations of past and present tenses, and how they might be used.

Here we have a box that prints out its current state and its history each time we open and close it:

```
"Tense Boxing"

The Temporal Prism is a room. "A room of angled
mirrors, in whose surfaces you can see what is now;
what just was; what has always been. A final mirror is
broken and its frame gapes blackly."

The mysterious box is in the Temporal Prism. It is an
openable closed container.

To assess the box:
    if the box was not open, say "The box was not open.";
    if the box was open, say "The box was open.";
    if the box had not been open, say "The box had not
been open.";
    if the box had been open, say "The box had been
open.";
    if the box is not open, say "The box is not open.";
    if the box is open, say "The box is open.";
    if the box has not been open, say "The box has not
been open.";
    if the box has been open, say "The box has been
open."
```

```
Before opening the mysterious box:
    say "You are about to open the box.";
    assess the box.

Before closing the mysterious box:
    say "You are about to close the box.";
    assess the box.

After opening the mysterious box:
    say "You now open the box.";
    assess the box.

After closing the mysterious box:
    say "You now close the box.";
    assess the box.
```

Note that "was..." and "was not..." and so on may describe conditions more complicated than simple properties: we could equally well ask "if the box has been in the sack", "if the box had been carried by the player", and so on.

The past ("if the box was...") and past perfect ("if the box had been...") are especially useful for cases where we want to report on an action after the state of the item has changed; so, for instance:

```
After taking the mysterious box:
    if the box had not been carried by the player, say
"You lift the mysterious box for the first time.";
    if the box had been carried by the player, say "You
again pick up the mysterious box."

Test me with "open box / close box / open box / take
box / drop box / take box".
```

This is in many respects similar to a rule beginning "After taking the mysterious box for the first time...", but it is superior in most

circumstances, for two reasons.

First, it will respond correctly even if the player has somehow carried the box before without taking it explicitly: for instance, if another character gave him the box, if the box were moved into his inventory as a result of another action, or if the player carried the box at the start of play. Inform begins its reckoning of time when the game begins, so if the box is defined as being open at the outset, "if the box has been open" will always be true.

Second, "after taking... for the first time" fires only the first time the player attempts to take something. If the player tried to take the box, failed, and then tried again later, the "for the first time..." rule would not fire; our "if the box has not been carried..." rule would.

---

150 **Example** **Bruneseau's Journey** *RB*

A candle which reacts to lighting and blowing actions differently depending on whether it has already been lit once.

> "Sire," said the Minister of the Interior to Napoleon,
> "yesterday I saw the most intrepid man in your Empire."
> - "What man is that?" said the Emperor brusquely, "and
> what has he done?" - "He wants to do something, Sire."
> - "What is it?" - "To visit the sewers of Paris."
>
> This man existed and his name was Bruneseau.
>
> - Victor Hugo, Les Miserables

Let's say that our intrepid explorer has a candle that can be lit and blown out again, and should accordingly appear unlit, burning, or partly burnt:

> "Bruneseau's Journey"

The Sewer Beneath St Denis is a room. "A narrow, stone-lined passageway, with only a little ledge to walk above the level of the refuse that flows down towards the Seine."

The candle is carried by the player. The description of the candle is "A candle, [if the candle has been lit]partially burnt[otherwise]still in pristine condition with untouched wick[end if]."

Instead of examining the lit candle, say "It burns with a pure heart."

The block burning rule is not listed in the check burning rules.

Instead of burning the lit candle:
    say "The candle is already lit."

Check burning:
    if the noun is not the candle, say "[The noun] cannot profitably be set on fire."

Carry out burning the candle:
    now the candle is lit.

Report burning:
    if the candle had been lit, say "You relight the candle.";
    otherwise say "You light the candle for the first time.".

Understand "blow out [something]" as blowing out.

```
Understand "blow [something] out" as blowing out.
Blowing out is an action applying to one thing.

Carry out blowing out the candle:
    now the candle is unlit.

Report blowing out:
    if the noun is the candle and the candle was lit, say
"You blow out [the noun].";
    otherwise say "You blow on [the noun], to little effect."

Test me with "x candle / light candle / x candle / blow
out candle / x candle".
```

We must be careful: "if the noun was lit" would throw errors because past-tense rules can only be applied to specific items, not to variables that could be anything.

---

151 ExampleElsie *RB*

A door that closes automatically one turn after the player opens it.

Suppose we want to create an automated door of the sort that closes when it isn't in use. A convenient way is to write a rule that fires "every turn when the sliding door was open". This will be true only if the door was open at the beginning of the turn: if the player just opened it this turn, it stays open, and if it was already closed, it stays closed. Thus:

```
"Elsie"

The axis-ward is a direction. The opposite of axis-ward
is hub-ward. Understand "aw" or "axisward" as axis-
ward.
The hub-ward is a direction. The opposite of hub-ward is
```

```
axis-ward. Understand "hw" or "hubward" as hub-ward.


The Ship's Bridge is a room.


The sliding door is a door. It is axis-ward from Bridge
and hub-ward from C Deck. The initial appearance is
"There is a door in this wall[if closed] -- or at least, the
potential for a door, since currently it is sealed,
distinguishable from the rest of the wall only by the
warning stripes on its surface[end if]."


Every turn when the sliding door was open:
    now the sliding door is closed;
    if the player can see the sliding door:
        say "The sliding door slips back into place, sealing
the wall as though there had never been a breach."


After opening the sliding door:
    say "You press the appropriate buttons, and a section
of wall slides away."


Test me with "open door / look / enter door / z".
```

ExampleInfiltration *RB*

A room whose description changes depending on the number of times the player has visited.

Suppose we have a location that makes the player uncomfortable, and we want its description to change slightly each time he goes there, to reflect his increasing unease. We also want the door to that room to show whether he is going there for the first time, or whether this is a repeat visit.

We start with an ordinary room:

"Infiltration"

The Wasteland is a room. "In its more distant reaches, the Wasteland has a kind of austere beauty, but here beside the Secure Zone it is the worst of all possible worlds. Barrels of toxins are stacked the regulation hundred and fifty feet out; more traditional garbage has simply been flung over the wall, and this category includes one or two corpses roughly and inadequately disguised by black plastic bags. The wall itself has become a canvas for outcasts and exiles, and is covered with obscene paintings, lewd remarks about the inhabitants of the Secure Zone, and a few maudlin epitaphs."

Now the door, which will change from saying "leads inside..." to "leads back inside..." when this becomes appropriate:

The portal is a door. It is inside from the Wasteland and outside from the Secure Zone. "[if the player is in the Wasteland]To the west, a[otherwise]A[end if] portal in the cinder-block and barbed wire wall leads[if the player is in the Wasteland and the Zone is visited] back[end if] [if the player is in the Wasteland]inside[otherwise]outside[end if]."

Here we haven't used any conditions that we didn't know about in previous sections: the portal line only reflects whether the Zone has been visited never or visited once. But the Secure Zone itself makes use of the number of times visited:

The Secure Zone has the description "[if the player is in the Zone for the second time]Re-entering the Zone has not made you any more comfortable inside. [end if]Despite your carefully-chosen outfit and the walk you have been practicing, you are sure those inside can tell you don't belong. Not that very many people are visible here[if the player is in the Zone for more than the second time] -- the place seems more desolate than ever[end if]."

```
Instead of going west in the Wasteland, try going inside.
Instead of going east in the Secure Zone, try going
outside.
```

And finally, to be sure that the player does see our fancy changing descriptions:

```
Use full-length room descriptions.
```

```
Test me with "look / open portal / w / look / e / look / w /
e / w".
```

Notice that the description of the Secure Zone changes from visit to visit, but that looking repeatedly during a single visit changes nothing.

---

153 **Example** **Annoyotron Jr** *RB*

A child who after a certain period in the car starts asking annoying questions.

```
"Annoyotron Jr"
```

```
The Minivan is a room. The Open Road is outside from
the Minivan. Pete is a man in the Minivan. "Pete [if the
player has been in the Minivan for 3 turns]is starting to
look bored[otherwise]is playing with his travel activity
book[end if]."
```

```
Every turn:
    if the player has been in the Minivan for 5 turns, say
"'Are we there [if saying no]now?'[otherwise]yet?' asks
Pete.[end if]"
```

```
Instead of saying no:
    say "'Oh,' says Pete. There is a blessed, momentary
silence."


Instead of going to the Open Road:
    say "You leap to your death.";
    end the story.


Test me with "z / z / look / g / g / g / no / z / z / z / no / z /
out".
```

Chapter 10: Scenes

10.4. During scenes; *§10.5. Linking scenes together*; *§10.6. More general linkages*; *§10.7. Multiple begi*

Contents of *Writing with Inform*
Chapter 9: Time
Chapter 11: Phrases
Indexes of the examples

§10.1. Introduction to scenes

As we have seen, Inform divides up space into individual places called "rooms", and allows us to group rooms together into "regions" if we find that convenient. And Inform also divides time up, into individual turns. These too we can group together: the equivalent of a region is a "scene".

To put this another way, if we think of the interactive fiction as a stage play, then up to now it has simply contained endless dialogue and stage directions - there has been no convenient way to divide up its running time into dramatic episodes, in the same way that a playwright might make Act II take place in the same drawing-room as Act I, but (let us say) six months later, after many things have changed. The script contains cues for one scene to end and another to begin: when those cues are reached, the stage hands rearrange props, actors reposition themselves and so on.

Inform also allows us to create scenes, with cues for them to start and end, and some stage machinery (so to speak) making it easy to move the action on. But interactive fiction is *interactive*, so the metaphor of the theatre only goes so far. We can have several different scenes going on at once - perhaps with the relevant events taking place in different rooms, which the player is free to walk between. And the player may make a choice which changes the story-line, causing scenes to happen which otherwise would not have happened, and so on. Scenes can even be "recurring", that is, can repeat themselves.

So organising the story-line into scenes is not simply a matter of making a list (Scene 1, then Scene 2, then Scene 3, *finis*). It is more like a chart in which one scene can lead in several possible ways to others - a sort of map of time, which as we shall see Inform displays in its "Scenes" index.

Start of Chapter 10: Scenes
Back to Chapter 9: Time: §9.15.

§10.2. Creating a scene

As usual, we only need to say that something is a scene to make it so:

```
Train Stop is a scene.
```

We conventionally write scene names with capital letters, as this demonstrates.

This works, and shows up in the "Scenes" index, but does nothing. We have given no instructions on when it begins - no cue, in stage-play terms - so it never will begin, and even if it did, nobody would notice since it does nothing. First, to give it a beginning:

```
Train Stop begins when the player is in the Station for the third turn.
```

In theory any condition can be used to cue the scene - here, it's "the player is in the Station for the third turn" - but it's wise to look for a state of affairs which will last at least a brief time, because scene changes only happen at the start and end of turns. (Something like "...when examining the timetable" may be true only for a part of the middle of a turn, and so go unnoticed.)

Every scene has two rulebooks attached, one at each end, so to speak. These look very like "when play begins" and "when play ends", and work in the same way. Thus:

```
When Train Stop begins:
    now the Flying Scotsman is in the Station;
    say "The Flying Scotsman pulls up at the platform, to a billow of
steam and hammering."
```

```
When Train Stop ends:
    now the Flying Scotsman is nowhere;
    if the player is in the Station, say "The Flying Scotsman inches
away, with a squeal of released brakes, gathering speed invincibly
until it disappears around the hill. All is abruptly still once more."
```

Thus when the scene begins, our imaginary stage-hands wheel in a steam train; when it ends, they get rid of it again. Note that we know where the player will be at the start of the scene, but by the end he may have wandered off across the fields, so we must be careful not to report something he might not be in a position to see.

When Train Stop begins, we printed some text, but we did this by hand. We didn't need to, because Inform automatically prints out the description of a scene (if it has one) when the scene begins. Scenes can have properties, just like objects, and in particular they have the "description" property. For example, we could write:

```
Arrival is a scene. "There's a flourish of trumpets."
```

which saves us the trouble of writing the rule:

```
When Arrival begins: say "There's a flourish of trumpets."
```

We can also write rules like this which apply to a whole variety of scenes at once. For instance:

```
A scene can be bright or dim. A scene is usually dim. Dawn is a
bright scene.

When a scene which is bright ends: say "So passes the bright [scene
being changed]."
```

Here, instead of naming a scene ("Train Stop"), we've given a description ("a scene which is bright"). When a scene begins, these general rules come before those which name the scene exactly; when it ends, the reverse is true.

Start of Chapter 10: Scenes

Back to §10.1. Introduction to scenes

Onward to §10.3. Using the Scene index

Example 154: ★ **Pine 1** Pine: Using a scene to watch for the solution of a puzzle, however

arrived-at by the player.

Example 155: ★★ **Entrapment** A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing situation.

---

§10.3. Using the Scene index

But when we test the previous section's example, we find that after a brief wait, the train pulls up: but it never goes away again. We have given instructions on how the scene ends, but not when it ends, and as a result the scene goes on forever once started.

Even with simple story-lines, and this one could hardly be simpler, it is surprisingly easy to overlook something so that the whole story-line is derailed.

The Scenes page of the index is intended to help with this. The Plot section shows all of the scenes and how they are to begin, along with a key to the symbols used on it. One scene always included is "Entire Game", a special scene which, as its name implies, is always being played out. But if we look at the Scene index for the previous example, we will also see our Train Stop scene, and find that it is marked with the red warning symbol for "never ends". Let us fix this:

```
Train Stop ends when the time since Train Stop began is 3 minutes.
```

Note the useful value "time since Train Stop began":

time since (scene) **began ... time**

This phrase produces the time since the named scene began, which only makes sense, of course, if it has indeed begun. Example:

```
time since Entire Game began
```

time since (scene) **ended ... time**

This phrase produces the time since the named scene ended, which only makes sense, of course, if it has indeed ended. Example:

```
time since Formal Dinner ended
```

The actual times, in case they are needed, can be obtained with:

time when (scene) **began ... time**

This phrase produces the time (i.e., the value of the "time of day" variable) at the moment when the given scene began.

time when (scene) **ended ... time**

This phrase produces the time (i.e., the value of the "time of day" variable) at the moment when the given scene ended.

The testing command SCENES causes Inform to monitor the beginning and ending of scenes, and report on them. For instance:

```
>ask monk about lodging
"Welcome a poor traveler for the night?" you ask, rubbing your
fingers together to restore a little feeling.

The monk looks you up and down for a moment and you sense his
reaction hanging in the balance; then he slaps you on the back, hard
enough to drive the air from your lungs. "In."

[Scene 'Greeting' ends]
The monk takes your elbow and pushes you imperiously toward
dinner.
[Scene 'Banquet' begins]
```

Start of Chapter 10: Scenes
Back to §10.2. Creating a scene
Onward to §10.4. During scenes

Example 156: ★ **Age of Steam**
The railway-station examples so far put together into a short game called "Age of Steam".

§10.4. During scenes

Scenes are not only useful for changing the setting, by moving items or people around and providing a little narration. We can also make the rules different in one scene from another. For instance, at a sleepy country halt there is no reason why one should not walk across the tracks: but if there is a train in the way, that would be impossible.

```
Before going north during the Train Stop, say "The train blocks your
way." instead.
```

Any rule can have the clause "during ..." attached, provided that clause goes at the end and either explicitly names a scene, or gives a description of which scenes would match. This is especially useful with "every turn":

```
Every turn during the Train Stop, say "Water is sluiced out of the tank
and into the engine."
```

We can test whether a scene is happening with the adjective "happening":

```
if Train Stop is happening, ...
```

if (scene) **has happened:**

This condition is true if the given scene has both begun and ended.

if (scene) **has not happened:**

This condition is true if the given scene has not ended (or never started).

if (scene) **has ended:**

This condition is true if the given scene ended at least once.

if (scene) **has not ended:**

This condition is true if the given scene has never ended.

We need to be a bit careful: it's possible to set things up so that the Train Stop scene will play out more than once, so "Train Stop is happening" and "Train Stop has happened" might both be true at once.

The kind of value "scene" is one which is allowed to have properties - it has a tick in the "properties" column in the chart in the Kinds index - and this can be very useful in describing scenes. For instance, we could write:

```
A scene can be thrilling or dull. Train Stop is dull.
A scene has a text called cue speech. The cue speech of Train Stop
is "All aboard!".
```

Inform has the adjectives "recurring", "non-recurring" and "happening" all built in to describe scenes, and the above would add "thrilling" and "dull". Moreover, the "during" clause of a rule can give a description of a scene as easily as a specific scene name. For instance:

```
Before going north during a dull non-recurring scene, ...
```

Start of Chapter 10: Scenes

Back to §10.3. Using the Scene index

Onward to §10.5. Linking scenes together

Example 157: ★ **Full Moon** Random atmospheric events which last the duration of a scene.

Example 158: ★★ **Space Patrol - Stranded on Jupiter!** We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

Example 159: ★★★ **Bowler Hats and Baby Geese** Creating a category of scenes that restrict the player's behavior.

Example 160: ★★★ **Day One** A scene which plays through a series of events in order, then ends when the list of events is exhausted.

---

§10.5. Linking scenes together

Let us suppose that somebody gets off the train, after all, so that a second scene follows on.

```
Brief Encounter is a scene. Brief Encounter begins when Train Stop
ends.
```

The effect of this is that they occur in sequence. If we add a third to the chain of scenes:

```
Village Exploration is a scene. Village Exploration begins when Brief
Encounter ends.
```

...we find another chance to fool ourselves: if we check the Scenes index again, we can see the linkages between these scenes, but we also see that Brief Encounter never ends (despite its name). All we have said is that another scene begins where Brief Encounter leaves off, but it never does, so this is moot.

Start of Chapter 10: Scenes

Back to §10.4. During scenes

Onward to §10.6. More general linkages

Example 161: ★★★ **Pine 2** Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

---

§10.6. More general linkages

We are allowed to link the beginning or end of any scene to the beginning or end of any other scene. So, for instance:

```
Luggage Trouble is a scene. Luggage Trouble begins when Brief
Encounter begins.
```

Thus the two scenes run concurrently, at least for a while. We can also add that:

```
Luggage Trouble ends when Brief Encounter ends.
```

This can be useful when a large, complicated scene really contains several smaller sub-scenes.

A special exceptional case is that we can have any scene or scenes starting right at the outset:

```
Railway Meeting is a scene. Railway Meeting begins when play
begins.
```

When play ends, of course, all scenes end, so there is no need to say that.

Start of Chapter 10: Scenes

Back to §10.5. Linking scenes together

Onward to §10.7. Multiple beginnings and repeats

Example 162: ★ **The Prague Job** Scenes used to provide pacing while the player goes through his possessions.

Example 163: ★★★ **Entrevaux** Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

§10.7. Multiple beginnings and repeats

It is quite allowed for a scene to be linked to several other scenes, and this is useful if several alternate strands of plot are being brought together in a common resolution scene:

```
Bittersweet Ending begins when Stranger's Rejection ends.
Bittersweet Ending begins when Stranger's Acceptance ends.
```

and we can also have the same scene beginning when a condition holds. In general, it will begin the first time it gets any chance to do so.

All scenes are ordinarily set up so that they can happen only once. But sometimes we want them to repeat. Suppose the train calls not once only, but every twenty minutes. We could set this up with two scenes linked back to back like so:

```
Train Stop is a recurring scene. Train Wait is a recurring scene.
Train Wait begins when play begins.
Train Stop begins when Train Wait ends.
Train Wait begins when Train Stop ends.
```

The difference here is that these scenes have been declared as "recurring". In all other respects they are the same as any other scene.

Start of Chapter 10: Scenes

Back to §10.6. More general linkages
Onward to §10.8. Multiple endings

Example 164: ★ **Night and Day** Cycling through a sequence of scenes to represent day and night following one another during a game.

Example 165: ★★★ **Pine 3** Pine: Allowing the player to visit aspects of the past in memory and describe these events to the princess, as a break from the marriage-proposal scene.

---

§10.8. Multiple endings

Interactive fictions vary considerably in the extent to which the player is allowed to influence the story-line. Sometimes the appearance of making choices is wholly bogus, as only one possible line is given more than lip service. But in other works, the player can radically change the outcome, and whole rafts of plot differ between one person's experience and another's. Inform allows multiple endings to its scenes to make this kind of branching story-line easy to achieve.

Any scene can have up to 31 alternate endings, differentiated by name (unless the Z-machine format has been selected on the Settings panel, in which case, 15). These alternates are created as and when conditions are set for them:

```
Brief Encounter ends happily when ...
Brief Encounter ends wisely but sadly when ...
```

"Ends happily" and "ends wisely but sadly" behave just like "ends". We can have rules "When Brief Encounter ends happily, ..." and so forth, in addition to rules "When Brief Encounter ends, ..." - if a rule doesn't specify any particular ending, it applies to all of them.

We can also link rules together from these branches, so

```
Stranger's Acceptance begins when Brief Encounter ends happily.
```

```
Stranger's Rejection begins when Brief Encounter ends wisely but
sadly.
```

With this set-up and that of the previous section, there are now two possible paths through the story:

```
(i) Train Stop - Brief Encounter - Stranger's Acceptance - Bittersweet
Ending
(ii) Train Stop - Brief Encounter - Stranger's Rejection - Bittersweet
Ending
```

We might later need to know which of these paths has been taken, and to help with this Inform provides conditions like so:

```
if Brief Encounter ended happily ...
if Brief Encounter did not end happily ...
if Brief Encounter ended wisely but sadly ...
if Brief Encounter did not end wisely but sadly ...
```

(For a scene which repeats, note that these conditions apply only to the most recent repetition: and that such conditions are always false if the scene is currently going on. "Brief Encounter did not end happily" will be true only when the scene has finished but in a different way.)

Start of Chapter 10: Scenes

Back to §10.7. Multiple beginnings and repeats

Onward to §10.9. Why are scenes designed this way?

Example 166: ★★★ **Panache** Replacing the score with a plot summary that records the events of the plot, scene by scene.

Example 167: ★★★ **Pine 4** Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the Princess has understood the point.

---

§10.9. Why are scenes designed this way?

In the part it plays in stories, time is like space. The endings of a scene (along with its beginning) are like the map connections leading out of a room. The Scenes index keeps track of the "map of time" through which these possible story-lines traverse. Some works of IF will have immensely complicated story-lines in only a few rooms, some will have no scenes at all despite a sprawling geography. The Scenes and World index tabs, side by side, show both kinds of map.

Just as Inform uses a simple but practical design for the boundaries between rooms (map connections and doors, that is), it also simplifies transitions between scenes. Scenes are based on states of things: we give circumstances for them to begin or end. There is no phrase with the power to say "make Act II begin right now", so perhaps it is worth explaining why not. The state-based approach was chosen because:

* it guarantees that each action falls entirely inside, or entirely outside, of any given scene (and therefore that "during..." clauses in the conditions for a rule are not affected by rule ordering);

* it ensures that scene changes occur outside actions, like every turn rules;

* it promotes a style of writing which makes it clearer to the reader of the source text when a scene begins and ends, and what conditions are guaranteed to be true during it;

* it makes it possible for the Scenes index page to show this information in a communicative way.

Settings in IF where one revisits the same location but at a different time, or after a dramatic change, have historically been difficult to test properly and prone to mistakes. (The classic example would be where a character killed during Act I reappears unharmed in Act II.) The design of scenes is an attempt to encourage a style of writing which minimises the risk of these accidents.

Since scenes are, in the end, only a convenient way to organise rules, and do nothing that cannot be done by other means, this simplified system of scene changing does not really restrict us.

Start of Chapter 10: Scenes
Back to §10.8. Multiple endings
Onward to Chapter 11: Phrases: §11.1. What are phrases?
Example 168: ★★★ **Cheese-makers** Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

Examples from Chapter 10: Scenes

Start of this chapter
Chapter 11: Phrases
Indexes of the examples

★ **154 ExamplePine 1** *RB*

Pine: Using a scene to watch for the solution of a puzzle, however arrived-at by the player.

Because scene rules are checked every turn, they can be useful for designing puzzles which have multiple solutions. Instead of deciding the puzzle is "solved" when the player does a certain action, we set up a scene that checks to see whether the player has achieved a certain outcome -- however he accomplished it.

For instance, in this scenario, we're waiting for Sleeping Beauty to wake up, and it doesn't much matter how...

```
"Pine"

A person can be asleep or awake. A person can be
active or passive.
```

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking a little confused[end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but how?"

Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"

```
Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a
terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour
[something] on [something]" or "splash [something] at/
on [something]" as throwing it at.

Test me with "x beauty / wake beauty / pour water on
beauty".
```

---

155 **ExampleEntrapment** *RB*

A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing situation.

The power of scenes lies in their ability to watch for general conditions and move the narrative along whenever these are fulfilled. Instead of waiting for the player to do one specific thing, the game waits for the world to be in a certain condition, before moving to the next stage of the plot.

For instance, suppose we have a story in which the player has been captured for doing something inappropriate at court and is brought in to await a meeting with a palace official. We want to give the player a few minutes to stew, and we want the scene to end with him doing something mildly peculiar or embarrassing, and the official catching him in the act. So we tempt him into trying any of a number of different kooky activities, and just wait until he falls into the trap...

"Entrapment"

Waiting Suite is a room. "You find yourself in a narrow room, more cozy than is really comfortable, with dark paneling on all the walls. Underfoot is a thick carpet the color of dried blood. The head of a dragon kit is mounted on the wall."

The wood paneling is scenery in the Waiting Suite. The description is "Just the sort of ornate panels that might conceal a carved switch. You've heard all sorts of rumors about secret rooms and passages in the palace, some of which have not been opened in centuries because no one remembers how to get at them." Understand "panels" or "panel" or "panelling" as the paneling.

Instead of switching on the paneling, say "First you'll have to locate any switches or catches with a careful search."

The thick carpet is scenery in the Waiting Suite. Understand "red" or "blood" or "rug" as the carpet. The description is "A dull, unwelcoming weave, only a touch redder than the wood around you. You discern that it does not lie perfectly flat."

Instead of touching the paneling for the first time: say "You run your hands over the paneling with a methodical touch, knowing exactly what you're looking for but never quite feeling anything that gives or twists; then thump lightly, looking for hollow spaces."

Instead of touching the paneling for the second time: say "With increased vigor, you run your fingers along the borders between panels, then smack each panel

sharply at the center. No luck yet, but if you keep at it, you're bound to turn up anything that's there to find."

Instead of attacking the paneling: try touching the paneling. Instead of searching the paneling: try touching the paneling. Understand "knock on [something]" or "tap [something]" or "tap on [something]" as attacking.

After touching the paneling when the player is not confident:
    say "Having polished off all the panels within easy reach, you now have to contort yourself around furniture here and crawl along the floorboards there, hitting each panel three times quite solidly before moving on.";
now the player is embarrassed.

Instead of looking under the carpet for the first time:
    say "You take a corner of the carpet and tug. The floor is sticky, so it doesn't come up on the first try."

A small table is an enterable supporter in the Waiting Suite. On the table is a copy of Dragon Pursuit Today. The description of Dragon Pursuit Today is "Full of glossy illustrations of dragons in various stages of capture, captivity, and destruction. The back of the magazine contains small black-and-white advertisements for hunting kits and the like." Some advertisements and some illustrations are part of Dragon Pursuit Today. The description of the illustrations is "You have the misfortune to look first at the photographs accompanying 'Cleaning Dragon Splanchna', and feel quite unwell." The description of the advertisements is "Mostly terse ads and phone numbers."

After looking under the table:
    say "It's quite a low table and you have to get down on your knees and poke your head underneath in order to get a good look."

After looking under the carpet:
    say "You pull again at the carpet. There is a tug, then a tearing, as the ancient fabric struggles against the fabric glue. Some of the carpet winds up in your hand and some of it remains in patchy threads adhering to the floor."

After entering the table:
    say "You climb onto the small table, noticing belatedly that you are leaving muddy footprints on its polished surface. Oh well: you can wipe them away again when you get down."

The dragon head is scenery in the Waiting Suite. Understand "kit" or "mouth" as the dragon head. The description is "Its eyes are wide with bewildered surprise; its mouth gapes, its forked tongue protrudes indignantly. From down here it looks as though there's something shiny stuck in its mouth, though you can't tell for sure." The head contains a shiny thing. The description of the shiny thing is "Intriguing but impossible to see clearly." Instead of taking the shiny thing, try searching the dragon head.

Before searching the dragon head:
    if the player is not on the table, try entering the table;
    if the player is not on the table, stop the action.

After searching the dragon head: say "You have a good look inside the dragon's mouth. There's a ball of lucite inside, propping the jaw in display position."

A person can be confident, nervous, or embarrassed. The player is confident.

```
Touching the paneling is embarrassing behavior.
Looking under the carpet is embarrassing behavior.
Entering the table is embarrassing behavior. Looking
under the table is embarrassing behavior.



Instead of embarrassing behavior:
    if the player is nervous, now the player is
embarrassed;
    if the player is confident:
        say "Before you can act, you hear movement from
the inner office. You freeze, not quite ready to be
discovered in this situation. But no one comes out, and
you begin to breathe more easily.";
        now the player is nervous;
    otherwise:
        continue the action.



Causing trouble is a scene. Causing trouble begins
when play begins. Causing trouble ends when the
player is embarrassed. When Causing trouble ends: say
"Just at this inopportune moment, you hear a throat
being cleared behind you. 'We can see you now within,'
says a dry voice."; end the story saying "To be
continued..."



Test me with "switch paneling / touch paneling / g / g /
g".



Test more with "x dragon / x shiny / search head / g".
```

...and this scene might lead to another, and so on.

The purpose of an open-ended scene like this might be puzzly or narrative: we might be waiting for the player to get a puzzle solved, or we might be waiting for him to fulfil some plot condition that must be met before we can go on.

Example **Age of Steam** *RB*

The railway-station examples so far put together into a short game called "Age of Steam".

The following source is very short and simple, yet it already feels surprisingly interesting in play, because something is going on which the player does not control but must observe. The single scene both starts and finishes.

```
"Age of Steam"


The Station is a room. "Eynforme Halt is a raised
platform fringed with cowslip: a whistle-stop with no
more than a signal and a water-tank."


The Flying Scotsman is fixed in place. "The Flying
Scotsman, fastest train in the world, is now at a dead
standstill."


Train Stop is a scene. Train Stop begins when the
player is in the Station for the third turn. Train Stop ends
when the time since Train Stop began is 3 minutes.


When Train Stop begins:
    now the Flying Scotsman is in the Station;
    say "The Flying Scotsman pulls up at the platform, to
a billow of steam and hammering."


When Train Stop ends:
    now the Flying Scotsman is nowhere;
    say "The Flying Scotsman inches away, with a squeal
of released brakes, gathering speed invincibly until it
disappears around the hill. All is abruptly still once
more."
```

Instead of entering the Flying Scotsman, say "Alas, the [time when Train Stop began] arrival is only to take on water, not to set down or pick up."

Test me with "z / z / z / enter flying scotsman / z / z".

Example **Full Moon** *RB*

Random atmospheric events which last the duration of a scene.

"Full Moon"

Wolf Pursuit is a scene. Wolf Pursuit begins when play begins.

Every turn during Wolf Pursuit, say "[one of]A twig snaps behind you![or]The wind howls in your ears.[or]You feel chilly.[at random]".

When play begins:
    say "You have lost your spectacles, and the lamp, and can see barely further than the next tree. Roots keep trying to trip you, too..."

The Dark Forest is a room. "You are mostly aware that you are not as alone here as you would like, and that the ground is uneven."

Test me with "z / z / z / z".

 Example **Space Patrol - Stranded on Jupiter!** *RB*

We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

American radio adventure series of the 1950s were unobtrusively sponsored by breakfast cereals, as the following modest example demonstrates. Note that the scene-changing for the commercial break needs to know nothing about the actual programme it breaks into: if Part I were replaced with a different Space Patrol episode, Part II need not be changed at all.

"Space Patrol #57 - 1953-10-31 - Stranded on Jupiter!"

Use scoring.

Part I - Serial

Red Spot is a room. "You are in the middle of a vast red oval plain. Overhead, the thick Jovian clouds swirl menacingly, and a fine acrid dust falls instead of rain." Some acrid dust is scenery in the Red Spot. The description of the dust is "The rust-colored dust coats every surface. You've no idea how deep it goes."

Instead of going in Red Spot, say "As you once told Cadet Lucky, Jupiter's a mighty big planet, maybe bigger than Iowa. Why, the Red Spot alone stretches out almost to the horizon."

The player wears a silver uniform and rubber boots. The

player carries a shovel and an Analscope. The description of the Analscope is "As you recall from Space Patrol #9 - 1952-11-29 - The Electronic Burglar, the Analscope is a device for locating buried metals. That's what guided you all the way from the orbit of Uranus. (Oh, all right, Neptune.) If only you hadn't crashed!"

The metal plate is a fixed in place container. It is openable and closed. In the metal plate is some water. The description of the metal plate is "Stamped with the distinctive logo of the previous mission."

Instead of examining the player, say "Your hair clumps together stickily, thanks to the dustfall."

Digging is an action applying to one thing. Understand "dig [something]" or "dig in [something]" as digging.

Instead of digging the dust, try looking under the dust. Instead of looking under the dust when the metal plate is not visible: move the metal plate to the location; say "You brush aside the dust underfoot and -- what were the odds? -- it turns out that you landed just where the previous landing party did, thirteen ill-fated years ago. Here is the metal plate that covers their original well.

But wait! Called by the clanging of your shovel on the plate, a band of Jovian pterodactyls swoop down to attack! You're totally defenceless! You don't have a hope! You're absolutely finished!"; increase the score by 10; move K-Klak to Red Spot.

K-Klak the Pterodactyl is an animal. "K-Klak, leader of the Jupiter Pterodactyls, menaces you. A terrifying creature of scaly wings, with a dragon's tail, K-Klak

stands... about 1/8th of an inch tall." Instead of doing something to K-Klak, say "K-Klak makes a frankly panicky noise and leaps backwards, out of your way."

After opening the metal plate: increase the score by 10; say "You have found water! You're saved! K-Klak makes a (very cautiously) pleased noise. Now to find the stolen Brainograph, and track down the crook with the thick Jewish accent and his henchmen with their thick Polish accents..."; end the story finally.

The maximum score is 20.

Part II - Cereal

When play begins, say "Instant Ralstons and Regular Ralstons, the hot whole-wheat cereals in the red and white checkerboard packages present... SPACE PATROL... High adventure in the wild vast reaches of space... Missions of daring in the name of interplanetary justice... Travel into the future as Buzz Corey, Commander-in-Chief of the..."

Last score is a number that varies. Every turn: now the last score is the score.

Ralstons Ad is a scene. Ralstons Ad begins when score is not the last score. Ralstons Ad ends when the Ricechex is consumed.

Include Basic Screen Effects by Emily Short.

```
When Ralstons Ad begins:
    center "*** We'll be back in just a moment! ***";
    pause the game;
    strip the player;
    move the player to the Kitchen.


When Ralstons Ad ends:
    center "*** And now, back to today's exciting
adventure ***";
    pause the game;
    restore the player.


Saved location is a room that varies. Locker is a
container. Wardrobe is a container.


To strip the player:
    now every thing carried by the player is in the locker;
    now every thing worn by the player is in the
wardrobe;
    now saved location is location.


To restore the player:
    now every thing carried by the player is in the
Kitchen;
    now every thing in the locker is carried by the player;
    now every thing in the wardrobe is worn by the
player;
    move the player to saved location.


The Space Patrol Kitchen is a room. "The nerve center
of the Space Patrol! This is where cadets fill up with
their SUPER-FUEL. North leads to the astro control
room, while back south is the cargo hold." A breakfast
bowl is in the Kitchen. In the bowl is Ricechex. Ricechex
is edible. The Ricechex can be consumed or uneaten.
The Ricechex is uneaten.
```

Instead of going north in Kitchen: say "[refusal to leave]". Instead of going south in Kitchen: say "[refusal to leave]".

Instead of examining the player when Ralstons Ad is happening: say "You are currently being played by a generically attractive person of about 30, with very good teeth and well-kept nails."

After eating the Ricechex: say "That's right folks, always start your day the SPACE PATROL way with a tasty bowl of Ricechex, Wheatchex or good hot Ralstons. Mmmm Mmmm. You just can't get enough of the sugary goodness in Ricechex, Wheatchex and good hot Ralstons."; now the ricechex is consumed.

Instead of tasting the Ricechex:
    say "Wow! *wolf-whistle* Man oh man oh man! Yumm-y!"

To say refusal to leave:
    repeat through Table of Refusals:
        say "[nope entry][paragraph break]";
        blank out the whole row;
        rule succeeds;
    say "You can't. Eat your Ricechex."

Table of Refusals
nope
"You can't go that way in the limited universe of this sponsored message."
"Or that way."
"You've already tried that!"
"Why would you want to walk away when you have an alluring bowl of Ricechex right here?"

```
Test me with "n / i / x me / x dust / dig dust".

Test ad with "n / s / n / s / n / i / x me / get bowl / taste
ricechex / eat ricechex".

Test ending with "x plate / x k-klak / open plate".
```

Episode 57 of "Space Patrol" was actually called "Iron Eaters Of Planet X", just in case the reader feels that any of the foregoing unfairly traduces a work of thoughtful science fiction.

---

159 **Example** **Bowler Hats and Baby Geese** *RB*

Creating a category of scenes that restrict the player's behavior.

Scenes can have properties -- a fact that is very useful when it comes to writing a series of scenes that all need to act alike in some respect.

Suppose we have a plot that features a number of scripted scenes, where we need the player to stand still and wait while the events of the scene play out. One way to set this up is to create a property for such scenes -- let's call them "restricted" -- and then write a rule that keeps the player in place while the scene happens:

```
"Bowler Hats and Baby Geese"

Section 1 - The Procedure

A scene can be restricted or free.

Instead of going somewhere during a restricted scene:
    say "Better to stay here for the moment and find out
```

what is going to happen next."

And now let's set up our restricted scene. In it, a clown is going to turn up wherever the player is (it doesn't matter where on the map he's gotten to at this point) and do a performance; the player will not be able to leave the area until the performance completes. We'll start with the setting:

Section 2 - The Stage and Props

The Broad Lawn is a room. "A sort of fun fair has been set up on this broad lawn, with the House as a backdrop: it's an attempt to give local children something to do during the bank holiday. In typical fashion, everyone is doing a very good job of ignoring the House itself, despite its swarthy roofline and dozens of blacked-out windows."

The House is scenery in the Broad Lawn. The description is "A cautious vagueness about the nature of the inhabitants is generally considered a good idea. They might be gods, or minor demons, or they might be aliens from space, or possibly they are embodiments of physical principles, or expressions of universal human experience, or... at any rate they can run time backward and forward so it warbles like an old cassette. And they're always about when somebody dies. Other than that, they're very good neighbors and no one has a word to say against."

Instead of entering the House:
    say "You can't go in, of course. It's not a house for people."

The Gazebo is north of the Broad Lawn. "The gazebo is sometimes used for bands, but at the moment has been appropriated for the distribution of lemonade."

The clown is a man. "A clown wearing [a list of things worn by the clown] stands nearby." The description is "He winks back at you."

The clown wears a purple polka-dot bowler hat. He carries a supply of baby geese. The description of the supply of baby geese is "Three or four. Or five. It's hard to count." Understand "goose" or "gosling" or "goslings" as the supply of baby geese.

There are some eggs. The description of the eggs is "A blur, really."

There is a Spanish omelet. The description of the Spanish omelet is "Exquisitely prepared."

...And now the scene itself:

Section 3 - The Scenes

The Clown Performance is a restricted scene. Clown Performance begins when the turn count is 3.

When Clown Performance begins:
    move the clown to the location.

Every turn during Clown Performance:
    repeat through the Table of Clowning:
        say "[event description entry][paragraph break]";
        blank out the whole row;
        stop.

When Clown Performance ends:
    now the eggs are nowhere;
    now the clown carries the omelet.

Clown Performance ends when the number of filled rows in the Table of Clowning is 0.

Table of Clowning
event description
"A clown with a purple polka-dot bowler hat strides into the vicinity and begins to juggle baby geese."
"While the clown juggles, the baby geese visibly grow older and larger. The clown becomes unnerved."
"In an attempt to resolve the problem, the clown reverses the direction of his juggling. The geese revert to goslings."
"The goslings become smaller and smaller until the clown is juggling goose eggs[replace eggs]."
"The clown throws all the eggs into the air at once and catches them in the bowler hat. He takes a bow; the audience applauds. As a final gesture, he upends his hat to release a perfectly cooked omelet."

To say replace eggs:
    now the supply of baby geese is nowhere;
    now the clown carries the eggs.

Free Time is a scene. Free Time begins when Clown Performance Ends.

Test me with "scenes / n / z/ z / look / x geese / s / x geese / x eggs / z / s".

Example**Day One**
A scene which plays through a series of events in order, then ends when the list of events

```
"Day One"


Lecture is a scene. Lecture begins when play begins.


Every turn during Lecture:
    repeat through Table of Lecture Events:
        say "[event entry][paragraph break]";
        blank out the whole row;
        rule succeeds.
```

Here we use a table (see subsequent chapters) to keep track of all the events we wish to have occur during the course of the scene.

```
Table of Lecture Events
event
"'Welcome to Precolumbian Archaeology 101,' thunders Dr Freitag from the front of the class. 'Miss-- yes, you in the back. If you can't find a free seat, how are you going to find Atlantis? Sit down or leave. Now. Thank you.'"
"Freitag stands behinds his desk and lines up the pile of books there more neatly. 'It has come to my attention over previous years that there are two sorts of person who enroll in my class,' he says.


'Some of you will be members of the swim team or women's lacrosse players who have a distribution requirement to fulfill and are under the mistaken impression that archaeology must be easier than psychology. If that description applies to you, I advise you to drop the class now rather than at the midterm break. Under absolutely no circumstances will I ever sign a withdrawal form for someone who is crying at the time. Make a note of that, please.'"
"'The second sort of person,' Dr Freitag says, getting
```

another wind. 'Yes, the second sort of person takes this class because she imagines that it is going to lead to adventure or possibly to new age encounters with dolphins.'

His eye moves over the class, lingering an especially long time on a girl in a patchwork skirt.

'You should also leave now, but since you are probably lying to yourself about the reasons you're here, you will probably not heed my warning and we will be doomed to a semester of one another's company nonetheless.'"
"'Whatever you may tell yourself, you are not here to gain a deeper understanding of the world or get in touch with yourself or experience another culture.'

He paces before the first row of desks, hammering on them one at a time. 'I know you probably wrote an admissions statement saying that that is what you hoped to do. Well, too bad. It is not inconceivable that some of you, somehow, will muddle towards a deeper understanding of something thanks to this class, but I am not holding my breath, and neither should you.'"
"Freitag takes a breath. 'No, my dear freshwomen, what you are here to do is learn facts. FACTS. Facts are unpopular in this university and, I am unhappily aware, at most of the institutions of inferior preparation from which you have come. Nonetheless, facts it will be. I will expect you to learn names. I will expect you to learn dates. I will expect you to study maps and I will expect you to produce evidence of exacting geographical knowledge on the exams. I will expect you to learn shapes of pottery and memorize masonry designs. There are no principles you can learn which are more important or more useful than a truly colossal bank of facts right there in your own head.'"
"'I do not ever want to hear that you do not need to learn things because you will be able to look them up. This is the greatest fallacy of your computer-semi-literate generation, that you can get anything out of Google if you need it. Not only is this demonstrably false, but it overlooks something phenomenally important: you only know to look for something if you already know it EXISTS. In short there is no way to fake knowledge,

```
and I am not going to pretend there is.' He smiles in
lupine fashion.
```

```
'This class is likely to be the most miserable experience
of your four years in university. Clear?'"
"Everyone is silent."
"The lecture is interrupted by the shrill of a bell."
```

And then we define the scene so that it ends when the table runs out.

```
Lecture ends when the number of filled rows in the
Table of Lecture Events is 0.
```

One advantage of this is that we can then edit the events in the scene by changing just the table; the scene will always run the right length and end on the turn when the last event occurs.

And to add a few additional details:

```
Instead of doing something other than waiting, looking,
listening or examining during Lecture:
    say "Dr Freitag glares at you so fiercely that you are
frozen into inaction."
```

Notice the careful phrasing of "doing something other than..." so that we do not mention the objects; if we had written "something other than listening to something...", the instead rule would match only action patterns which involved a noun. We state the rule more generally so that it will also match nounless commands such as JUMP and SING, since Freitag will probably take a dim view of those as well.

```
When Lecture ends:
    now Freitag is nowhere;
    say "There is a flurry of movement as your fellow
students begin to put away their books. Dr Freitag
makes his way to the door and is gone before anyone
can ask him anything."
```

```
The Classroom is a room. Dr Freitag is a man in the
```

Classroom. "Dr Freitag paces before the blackboard."

Test me with "listen / x dr / x me / jump / z / z / z / z / z / x dr".

---

161 **ExamplePine 2** *RB*

Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

"Pine"

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking a little confused[end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

```
When Discovery ends: say "Throughout the palace you
can hear the other sounds of stirring and movement as
the spell of centuries is broken."


Instead of waking an awake person: say "Redundant."


Instead of waking an asleep person: say "Yes, but
how?"


Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much
for that true love's kiss nonsense."


Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"


Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].


[The second noun] wakes, shuddering. 'Agh! I had a
terrible dream about drowning and then-- Hey!'"


The player carries a jug of water. Understand "pour
[something] on [something]" or "splash [something] at/
on [something]" as throwing it at.
```

So much, we had before. Now, suppose we want a conversation style which allows the player to move conversation forward by asking appropriate questions, but which will keep moving forward even if he doesn't. To this end, we provide a table -- a borrowing from a later

chapter. In the table, we record two ways of performing each conversation bit, one which reflects the player's participation, and one in which the character moves things onward:

Table of Conversation

ask about (topic)	tell about (topic)
"Sleep/dreams/nightmares/sleep"	"Sleep/dreams/nightmares/sleep"
"Oh really, she's engaged? Your poor boy."	"Actually, she rejected me, leaving me a wreck. You could do much better, you know."
"Sorry, you say? This is a little weird, isn't it?"	"Sorry, you say? This is a little weird because we just met, but, um. Would you like to get married?"
"Sorry, you say? How rude."	She looks at you. "Do I suppose?"
"I was told you were going to marry me and live in the kingdom," you say, apologetically. "Would that be very bad?"	"I was told you were going to marry me and live in the kingdom," you say, apologetically. "Would that be very bad?"
'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.	'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.
"Do you think you could point out that it's been a hundred years since she last saw her boyfriend."	"Do you think you could point out that it's been a hundred years since she last saw her boyfriend."
You try to think how to point out that it's been a hundred years since she last saw her boyfriend."	You try to think how to point out that it's been a hundred years since she last saw her boyfriend."
"You've been updated for a hundred years, you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'	"You've been updated for a hundred years, you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

She looks to the window and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

You shrug, uncomfortable."

```
Instead of asking an awake beauty about a topic listed
in the Table of Conversation:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.
```

The "now Beauty is passive" line prevents her from making any conversation of her own on a turn when we've spoken to her. This keeps the conversation from progressing too quickly.

```
Instead of telling an awake beauty about something: try
asking the noun about it.
```

```
Instead of asking an asleep person about something:
    say "[The noun] snores."
```

```
Marriage Proposal ends when the number of filled rows
in the Table of Conversation is 0.
```

```
Every turn during Marriage Proposal:
    if Beauty is active:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.
```

After we've generated any spontaneous conversation, we return her to her regular active state.

```
Every turn: now Beauty is active.

When Marriage Proposal ends: end the story saying
"This is going to take some explaining."

Test me with "x beauty / wake beauty / pour water on
beauty / ask beauty about sleep / z / ask beauty about
marriage".
```

Now we have a scenario in which the player can ask her some questions out of order if he really wants to, but the scene will not end until the basic conversation topics have been exhausted. If we wanted to add some other chit-chat, not as part of the main conversation strand, but by way of optional enrichment, we might make a second conversation table and record alternative outcomes in it.

ExampleThe Prague Job *RB*

Scenes used to provide pacing while the player goes through his possessions.

Suppose we want to remind the player that he doesn't have all the time in the world, by starting to nag him when he's nearly, but not entirely, done going over his inventory in preparation for a job.

```
"The Prague Job"

A thing can be seen or unseen. A thing is usually
unseen. Carry out examining: now the noun is seen.
```

The player carries a lockpick, a smoke bomb, a grappling hook, and a pair of gloves. The description of the lockpick is "Effective on most kinds of key locks, it is a gift from your mentor in the discipline, old Wheezy." The description of the smoke bomb is "Your last of these, so you should rely on it only when other modes of escape have vanished. It takes effect when dropped, producing a cloud of purple haze sufficient to fill a medium-sized room." The description of the grappling hook is "Good for shooting at balconies and other sorts of overhang." The gloves are wearable. The description of the gloves is "Black and shiny, with gripping material on the palms. Batman would be jealous."

The Toilet is a room. "The walls are painted an unattractive green; the fixtures are a bit old. But it is the only place in the hostel with any privacy." The Long Hallway is outside from the Toilet.

Reviewing Possessions is a scene. Reviewing Possessions begins when play begins.

Escalating Danger is a scene. Reviewing Possessions ends when Escalating Danger begins. Escalating Danger begins when preparations near completion.

To decide whether preparations near completion:
    if at least two of the things which are carried by the player are seen, yes;
    no.

When Escalating Danger begins: say "Someone pounds on the door of your hideout and yells at you in Czech."

Instead of going from the Toilet during Reviewing Possessions: say "You need to go over your equipment

first, and make sure you're ready here."

Instead of going from the Toilet during Escalating Danger: say "You're not done checking over your materials."

Instead of waiting during Escalating Danger: say "There's no time to waste."

Every turn during Escalating Danger: if the time since Escalating Danger began is greater than 1 minute, say "Impatient footsteps pass your door again."

Escalating Danger ends when every thing which is carried by the player is seen. When Escalating Danger ends, say "There -- nothing damaged or torn. You're ready to go."

Mission is a scene. Mission begins when the player is in the Long Hallway. When Mission begins: end the story saying "The game is afoot"

Test me with "i / x lockpick / out / x bomb / out / x hook / x gloves / out".

ExampleEntrevaux *RB*

Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

For some games, it makes sense to organize the entire game around scenes rather than around locations, moving the player when a new

scene begins and laying out new props.

To this end, we might extend Inform's default handling of scenes so that each scene has properties to indicate prop lists and locations, and move objects in and out of play automatically as the scenes change. For instance:

```
"Entrevaux"

Part 1 - Procedure

A scene has a room called the starting location.

A scene has a list of objects called the scenery props.

A scene has a list of objects called the inventory props.
```

The starting location is the room to which the player should be moved; scenery props are things that need to be put there when the scene begins; inventory props, things that are given to the player when the scene begins; and the description some printed text to introduce the new scene. We may still occasionally need to have recourse to special "When the Dancing-Lesson begins..." rules for individual scenes, but for the most part this allows us to set scenes up in a consistent and predictable way.

Another point that might be slightly less obvious: sometimes we want to announce a change of location to the player when the scene starts, and sometimes we don't. In particular any scene that starts "when play begins" should probably not explicitly describe the entered room, since that would duplicate the description automatically produced on the first turn of play. So we add a property to track whether any given scene should be announcing its location:

```
A scene can be location-silent or location-loud.
```

And let's say that we also want to announce each new scene as another "chapter" of the game in play, with a pause before the scene begins.

Here we include "Basic Screen Effects" because it will allow us to pause the game for a keypress, then clear the screen before each new chapter:

```
Include Basic Screen Effects by Emily Short.

The chapter counter is a number that varies.

First when a scene (called the current scene) which is
not the Entire Game begins:
    if chapter counter is greater than 0:
        pause the game;
    increment chapter counter;
    say "[bold type]Chapter [chapter counter]: [current
scene][roman type]";

Last when a scene (called the current scene) which is
not the Entire Game begins:
    repeat with item running through the scenery props of
the current scene:
        move the item to the starting location of the current
scene;
    repeat with item running through the inventory props
of the current scene:
        move the item to the player;
    if the location is not the starting location of the current
scene:
        if the current scene is location-loud:
            move the player to the starting location of the
current scene;
        otherwise:
            move the player to the starting location of the
current scene, without printing a room description.
```

At the end of each scene, we strike the set and remove all the loose objects from play.

When a scene (called the current scene) ends:
    repeat with item running through things which are not fixed in place:
        if the item is not the player:
            now the item is nowhere.

Part 2 - Scenario

Entrevaux Station is a room. "The station building consists of a waiting room and a ticket-selling office so small that only one person can buy a ticket at a time. On the outside wall is a clock that runs twelve minutes late; but since the trains also run twelve minutes, give or take, behind their published schedule, this clock is helpful in establishing reasonable expectations. [paragraph break]Painted on the door is the logo of the Chemin de Fer de Provence, the only railway in France that is not part of the SNCF."

The Hillside Tower is a room. "It's very dark in here, lacking artificial lighting, but from the rough rectangular window you can see a slice of hillside and a little of the river Var."

The window is scenery in the Hillside Tower. The description is "Through it can be seen a slice of wooded hillside and exposed grey-brown cliff. You are in the southern French foothills of the Alps, and the territory is dry. The only respite is the river Var, a milky blue at this time of year, running shallowly over mud and large stones far below your window." Understand "view" or "slice of hillside" or "hillside" or "hill" or "river" or "var" or "mud" or "stones" or "large stones" as the window.

A used ticket is a thing. The description is "A piece of receipt paper indicating that you have paid the one-way fare of 9 euros from Nice. There is a hole punched through it."

A one-euro coin is a thing. The description is "It's a bimetal coin, brassy around the rim and silvery in the center. One side shows western Europe, with unusual prominence given to the UK, and the other side Leonardo da Vinci's four-armed, four-legged man having a nice stretch. It's dated 2002."

Some re-enactors are a person. "Milling about one end of the station is a crowd of medieval re-enactors." The description is "They're dressed in a somewhat aimless range of styles roughly honoring the period of 900-1500 AD. One gentleman is wearing a knobby leather cap; which is a good thing, because there is a rooster standing on his head." Understand "men" or "man" or "gentleman" or "rooster" or "reenactors" or "crowd" or "medieval" or "woman" or "women" as the re-enactors.

A kidnapper is a person. "Your kidnapper is watching you from the corner with his arms folded. You have the impression he's just marking time until someone more important arrives." The description is "He does not look at all like the kidnapping sort, but more like a sommelier at a superior restaurant: he wears a black pinstriped suit and has nicely-manicured hands."

The trolley is an enterable fixed in place container. "The 'train' on which you arrived is really just a single car, more like a trolley than a proper train." Understand "car" or "train" as the trolley. The description is "It has a glass front, so you can see ahead while riding: an innovation among trains."

Arrival is a location-silent scene. "After many days[']journey, you have arrived at last in Entrevaux, a walled medieval town now chiefly of interest to tourists and crusade re-enactors."

The starting location of Arrival is the Entrevaux Station.

The scenery props of Arrival are { re-enactors, trolley

}.
The inventory props of Arrival are { the used ticket, one-euro coin }.

Arrival begins when play begins. Arrival ends when the time since Arrival began is 2 minutes.

Abduction is a location-loud scene. "You check into the Hotel Vauban and sleep deeply enough; it was a long and sticky trip to get here.

Then in the middle of the night something confusing happens. You have the impression of strangers in your room, and then a searing pain, and you don't come back to yourself until midmorning of the following day..."
The starting location of Abduction is the Hillside Tower.
The scenery props of Abduction are { kidnapper }.

Abduction begins when Arrival ends.

Test me with "i / x re-enactors / z / z / i / x him".

ExampleNight and Day *RB*

Cycling through a sequence of scenes to represent day and night following one another during a game.

Suppose we want to have a sequence of nights and days in our game, with one scene to govern each daylight condition.

"Night and Day"

```
The sun is a backdrop. It is everywhere. The description
is "Currently out of sight."
```

```
Night is a recurring scene. Night begins when play
begins. Night begins when Dusk ends. Night ends when
the time since Night began is 3 minutes.
```

Notice that our two conditions for the beginning of Night are not in conflict: it will be night-time when the game begins, and then night will also recur every time the Dusk scene ends.

```
When Night begins:
    say "The sun falls below the horizon and the
temperature drops abruptly to well below zero.";
    now the description of the sun is "Currently out of
sight."
```

```
Dawn is a recurring scene. Dawn begins when Night
ends. Dawn ends when the time since Dawn began is 1
minute.
```

```
When Dawn begins:
    say "The sun appears on the horizon.";
    now the description of the sun is "It is tiny and weak.".
```

```
Day is a recurring scene. Day begins when Dawn ends.
Day ends when the time since Day began is 3 minutes.
```

```
When Day begins:
    say "The sun is now properly up."
```

```
Dusk is a recurring scene. Dusk begins when Day ends.
```

```
Dusk ends when the time since Dusk began is 1 minute.


When Dusk begins:
    say "The sun has passed across the sky and is on
the verge of setting."


Cratered Landscape is a room. "The ground here is [if
Night is happening]dim silver, with the craters visible as
darker splotches[otherwise]the color of dried blood; here
and there it is also rippled by impact craters[end if]. The
horizon curves visibly."


Test me with "z / z / z / look / x sun / z / z / z / z / z / z /
z".
```

If we run this example and then have a look at the scenes index, we'll see that the cycle is listed through thus:

```
Night (recurring)
    Dawn (recurring)
        Day (recurring)
            Dusk (recurring)
                Night
```

with the second "Night" in italics, to indicate that it is a repetition of the same scene that has already been listed above.

Example Pine 3 *RB*

Pine: Allowing the player to visit aspects of the past in memory and describe these events to the princess, as a break from the marriage-proposal scene.

```
"Pine"
```

Part 1 - The Set-up

This is mostly a repeat of what we have already seen, but for the sake of producing a playable scenario, we include it. The new material appears at Part 2.

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking [attitude][end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."; now Beauty is passive.

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but

how?"

Instead of attacking an asleep person:
now the noun is awake;
say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
now the noun is awake;
say "[The noun] slowly stirs to wakefulness!"

Instead of throwing water at an asleep person:
now the second noun is awake;
now the noun is nowhere;
say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour [something] on [something]" or "splash [something] at/on [something]" as throwing it at.

Table of Conversation

topic	reply
"Sleep/dream/nightmare/nightmares/sleep"	"Sleep well? You ask me, I'd say you had nightmares."
"Nothing/sleep/dream/nightmare/nightmares"	"Oh really, then maybe you should go back to bed."
"Marriage/wife/husband"	"Sorry, I just met you, but, um. Would you like to get married?"

She looks at you hopefully. 'Do I suppose to marry you or something.'"

"I was told I was going to marry the dragon lover," you say, apologetically. 'Would that be very bad?' This could be awkward, considering your family circumstances -- you did promise your mother that everything would be better, after this --

'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.

"Do you think you could point out that he's been in a sleeping curse, and that's why he hasn't been around for a while, so he might be worried.'

You try to think how to point out that it's been a hundred years since she last saw her boyfriend. And try not to think how awkward things would be in your family if she refuses to marry you."

"You've been waiting for a hundred years," you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

She looks out the window and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

You shrug, uncomfortable."

Instead of asking an awake beauty about a topic listed

```
in the Table of Conversation:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.


Instead of telling an awake beauty about something: try
asking the noun about it.


Instead of asking an asleep person about something:
    say "[The noun] snores."


Marriage Proposal ends when the number of filled rows
in the Table of Conversation is 0.


Every turn during Marriage Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.


Every turn: now Beauty is active.


When Marriage Proposal ends: end the story saying
"This is going to take some explaining."
```

So far we haven't much of a chance to affect matters and make them better. Suppose we'd like to add an element to the conversation where we're allowed to tell Beauty about past events -- and explore them a bit; and if the first retelling doesn't go quite as planned, we're allowed to revisit these scenes to hit them with a bit more emphasis.

Part 2 - Flashbacks

```
Instead of asking an awake beauty about a topic listed
in the Table of Flashback Material:
    now Beauty is passive;
    say "[reply entry][paragraph break]".
```

```
A fact is a kind of thing. The family circumstances is a
fact. A fact can be known or unknown. A fact can be
current or past.
```

Once known, a fact remains known permanently -- this could be useful if we wanted to make some rules about how Beauty acts when she knows different information. By contrast, a fact is only "current" if it is the last thing discussed. Since a player can mention a fact over and over, he can make it "current" again and again, and thus reactivate the flashback.

```
Table of Flashback Material

reply
"[if family circumstances is unknown]I wish you'd give me some
thought to your family circumstances, you say, your jaw tensing. 'I
wouldn't ask if it weren't for my [family
circumstances]...'[otherwise]'I don't think you fully understand the
[family circumstances],' you say.[end if]"
```

```
After printing the name of a fact (called target): now the
target is current; now the target is known.
```

This "After printing the name..." rule will be explained later in the chapter on activities; for now, it is enough to know that whenever family circumstances is mentioned in the table of flashback material, this rule will automatically be called. Now the terms under which the flashback happens:

```
Poverty flashback is a recurring scene. Poverty
flashback begins when family circumstances is current.
When poverty flashback begins: strip the player; move
the player to the Woodcutter's Shack.
```

Note the "recurring" here: we want the player to be able to revisit this scene as needed.

The Woodcutter's Shack is a room. "Your family lives in a shack in the forest. There are holes in the roof, and in the winter the snow comes in -- rain, too, for that matter. The walls aren't very well-boarded, and don't keep out the wind, and even though you live in the middle of dense woods, you can never gather enough fuel to keep this place fully heated. And then there's the stench. Pigs wander freely in and out, and your three youngest brothers play on the floor."

Pigs are an animal in the shack. The pigs are scenery. The description is "They really are very grubby, dirty animals, but what's worse than that, the value of pigs has declined a lot over the last few decades. This is hard to explain to someone who has been out of touch with the world for a while, but keeping pigs for meat is a dubious prospect when there's less and less for them to forage on." Instead of smelling the pigs: say "They smell the way animals do, when they live among their own refuse."; increase the pity of Beauty by 2.

The brothers are a man. The description of brothers is "Hans, Franz, and Lukas. Twins and then the baby. So young, and growing up fatherless; and soon to be orphaned entirely, if your mother's health does not improve." Understand "brother" or "twin" or "twins" or "baby" or "franz" or "hans" or "lukas" as the brothers.

The untidy bed is scenery in the Shack. Mother is a woman on the untidy bed. The description of mother is "She is wasting away of a slow disease, her skin stretched tautly over bone. She hasn't been the same since your father left." On the bed is a folded letter.

The description of the letter is "Many times read over and creased, the letter explains how your father has

gone away with a wealthy countess and will not return. Your mother was not able to read it herself, of course, and had to have it explained to her by the parish priest. Now she keeps it by the bed and crumples it in her fits of delirium."

Instead of kissing or touching Mother for the first time:
    say "You place a gentle kiss on her feverish brow. She looks up at you, her oldest -- yes, never mind that bit -- with a look of sincere trust and admiration.

'You'll find a way through this for us,' she says, squeezing your fingers. 'I know you will.'"; increase the pity of Beauty by 3.

Instead of kissing Mother: say "You have no more heart-rending memories of affection to recount; that one incident will have to serve, for rhetorical purposes."

Instead of waiting in the Shack: say "The wind blows sharply through the walls."

Instead of attacking someone in the Shack:
    say "Though sometimes the conditions of your life make you grouchy and impatient, you would never dream of striking a member of your own family. But from time to time you do feel the temptation."

Beauty has a number called pity. After examining something in the Woodcutter's Shack, increment the pity of Beauty. After examining mother, increase the pity of Beauty by 2. After examining the letter, increase the pity of Beauty by 3.

```
Poverty flashback ends when waiting or the time since
poverty flashback began is five minutes.


When Poverty flashback ends:
    now family circumstances is past;
    say "...you finish describing the miserable
circumstances of your home life, and allow your
attention to return to the present.";
    restore the player;
    now Beauty is passive;
    if Beauty is sympathetic, say "'Oh dear!' she says.
'What a dreadful life!' She wrings her hands. 'No wonder
you are eager to improve your lot...! But --' Her brow
clears, a new thought occurring. 'You needn't marry me,
you know! We could arrange it differently! I am certain
that my father would give you a large reward, instead,
and then I would not be separated from my current
boyfriend!'";
    otherwise say "She makes a disgusted face, but she
doesn't seem nearly so heart-wrung as you had hoped
to make her. Tough audience, these modern
princesses."


Definition: Beauty is sympathetic if the pity of Beauty is
greater than 4.


To say attitude:
    if Beauty is sympathetic, say "distressed on your
behalf";
    otherwise say "a little confused".
```

And the following is the same as in the Space Patrol example as well: we need a systematic way to remove the player's possessions, then put everything back when the flashback is over:

```
Saved location is a room that varies. Locker is a
container. Wardrobe is a container.


To strip the player:
```

```
    now every thing carried by the player is in the locker;
    now every thing worn by the player is in the
wardrobe;
    now saved location is location.


To restore the player:
    now every thing carried by the player is in the
location;
    now every thing in the locker is carried by the player;
    now every thing in the wardrobe is worn by the
player;
    move the player to saved location.


Test me with "x beauty / wake beauty / pour water on
beauty / ask beauty about sleep / tell beauty about
poverty / smell pigs / x mother / x letter / kiss mother /
ask beauty about marriage / z / z".
```

Because we haven't changed the endings of the "Marriage Proposal" scene, there is still only one way for this scenario to work out; but at least now the player has the opportunity to alter Beauty's attitude a bit (or not) before the game is done.

ExamplePanache

RB

Replacing the score with a plot summary that records the events of the plot, scene by scene.

If we have a plot that branches and has multiple kinds of outcome, we might well want to assemble these into a plot summary in place of the more traditional score. One way to approach this is to build the scene information into a table, adding information when each scene ends.

We begin with a bit of setup:

```
"Panache"
```

```
The player is in a room called Beneath Roxane's
Balcony. Christian is a man in the Balcony. "Christian
stands in a spot of moonlight and tries to avoid too
obviously glancing at the shadows that conceal you."
The description of Christian is "Like you, Christian loves
Roxane. Unlike you, he is handsome enough to receive
her favor in return. He is the beauty to your brain."

Roxane is a woman in the Balcony. "Above you in the
night is Roxane." Roxane can be wooed, skeptical,
confused, or annoyed. Roxane is skeptical. The
description of Roxane is "The brightest, the most radiant
of women -- and in love with an utter fool."

Empty Street is a room. "No one is about at this hour, all
alone under a pale moon."

Telling someone about something is speech. Asking
someone about something is speech. Answering
someone that something is speech.
```

This next portion borrows from the Advanced Actions chapter to allow us to command Christian to do things:

```
A persuasion rule for asking Christian to try speech:
persuasion succeeds.

Carry out Christian answering someone that something:
    now Roxane is wooed;
    say "'[noun], [the topic understood].'"

Carry out Christian answering the player that something:
    say "Christian parrots your words back to you."
instead.
```

```
Carry out Christian telling a skeptical Roxane about
something:
    now Roxane is confused;
    say "Christian turns to [the noun]. 'I must tell you
about [the topic understood],' he says, and comes to a
halt, looking at you for further direction.

    Perhaps you'd better give him exact lines to say.
Surely he can't mess up an instruction like 'say hello to
Roxane.'" instead.

Carry out Christian asking a skeptical Roxane about
something:
    now Roxane is confused;
    say "'So,' says Christian nervously to [the noun]. 'Did
you know about [the topic understood]?' But Roxane
merely seems puzzled." instead.

Carry out Christian telling a confused Roxane about
something:
    now Roxane is annoyed;
    say "Christian begins rambling on witlessly about [the
topic understood]." instead.

Carry out Christian asking a confused Roxane about
something:
    now Roxane is annoyed;
    say "Christian puts another confused question about
[the topic understood]." instead.
```

And now we have enough material to begin writing the scenes:

```
Courting Roxane is a scene. Courting Roxane begins
when play begins. Courting Roxane ends in success
when Roxane is wooed. Courting Roxane ends in failure
when Roxane is annoyed.
```

When Courting Roxane ends in success:
    record "Seduction by Proxy" in the Table of Events;
    say "Roxane, deeply moved by this sentiment, invites Christian up to her balcony. He scrambles up the ivy and disappears into her bedroom; the last thing you hear is a girlish giggle from above.";
    now Roxane is nowhere; now Christian is nowhere;
    move the player to Empty Street.

When Courting Roxane ends in failure:
    record "Ruining Christian's Chances" in the Table of Events;
    say "Roxane sighs heavily and goes back into her room, slamming the door behind her.

'Thanks very much,' says Christian to you, striding off down the street.";
    now Roxane is nowhere; now Christian is nowhere;
    move the player to Empty Street.

Sulky Ramble is a scene. Sulky Ramble begins when Courting Roxane ends in success. Sulky Ramble ends when the time since Sulky Ramble began is 2 minutes. When Sulky Ramble ends: record "Wandering the Streets, Sulking" in the Table of Events.

Every turn during Sulky Ramble:
    say "You find yourself kicking fenceposts quite without thinking about it."

Smug Ramble is a scene. Smug Ramble begins when Courting Roxane ends in failure. Smug Ramble ends when the time since Smug Ramble began is 2 minutes. When Smug Ramble ends: record "Wandering the Streets, Exultant" in the Table of Events; say "Of course, you will regret this soon enough."

```
Every turn during Smug Ramble:
    say "You find yourself smiling fiercely at the moon."


To record (occurrence - text) in (target table - a table
name):
    choose a blank row in the target table;
    now the event entry is the occurrence.


Table of Events
event
"A Duel of Insults"
with 30 blank rows.


The plot summary rule is listed instead of the announce
the score rule in the carry out requesting the score
rules.


This is the plot summary rule:
    say "The Plot So Far: [paragraph break]";
    let act number be 0;
    repeat through the table of Events:
        increment act number;
        say " Act [act number]: [event entry][line break]".


Test me with "christian, ask roxane about love /
christian, say your breath smells like ripe taleggio to
roxane / score / z / z / score".
```

ExamplePine 4 *RB*

Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the Princess has understood the point.

Suppose in addition to our pathetic little family history, we have another secret to convey to the Princess, this one a little more peculiar. She

either gets it or she doesn't; once she gets it, we do not revisit that flashback, though it is still possible to keep visiting the poverty flashback.

"Pine"

Part 1 - The Set-up

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking [attitude][end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."; now Beauty is passive.

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but how?"

Instead of waiting in the presence of an asleep person (called snorer): say "You are alone with the sound of [the snorer] snoring sonorously."

Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"

Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour [something] on [something]" or "splash [something] at/on [something]" as throwing it at.

Table of Conversation

topic	reply
"Sleep/dream/nightmare/nightmares/sleep"	"Sleep, when you ask me directly, is a fickle thing.

'Oh really, it's nothing. You're so much prettier than her anyway. I mean, you look much better from this angle.'

'Sorry, I have to go. This is a little weird because I just met, but, um. Would you like to get married?'

She looks at you nervously. 'Do I have to? I mean, I'd rather not.'

'Well, this escalated quickly. I'm not supposed to marry you or something.' Uh oh.'

'I was just wondering if you'd be willing to rule the kingdom,' you say, apologetically. 'Would that be very bad?' This could be awkward, considering your family circumstances -- you did promise your mother that everything would be better, after this --

'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.

'Do you think you could point out that it's been a hundred years since she last saw her boyfriend. And try not to think how awkward things would be in your family if she refuses to marry you."

You try to think how to point out that it's been a hundred years since she last saw her boyfriend. And try not to think how awkward things would be in your family if she refuses to marry you."

'Darling, I love you, but I don't think you're going to be able to if he's no longer interested in you?' you suggest.

She gives you the look of a wounded squirrel. 'My father might not approve of my love for the kitchen boy, but his heart is faithful and true!' she exclaims.

"Right, supposing that he's still around, she says at last, with a look you won't know how to consider, considering your fingers. Maybe such a letter as my William writes about the kitchen, after all she might be more fair-minded than you'd think, without telling you this in there in her base-born romances."

Right then. Perhaps you'd better tell her your secret, in exchange?"

"You've been up there for a thousand years," you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

She shakes her head and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

You shrug, uncomfortable."

Instead of asking an awake beauty about a topic listed in the Table of Conversation when Marriage Proposal is happening:

now Beauty is passive;
say "[reply entry][paragraph break]";
blank out the whole row.

Instead of telling an awake beauty about something: try asking the noun about it.

```
Instead of asking an asleep person about something:
    say "[The noun] snores."
```

```
Marriage Proposal ends in failure when the number of
filled rows in the Table of Conversation is 0.
```

```
Every turn during Marriage Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.
```

```
When Marriage Proposal ends in failure: end the story
saying "This is going to take some explaining."
```

Part 2 - Flashbacks

```
Instead of asking an awake beauty about a topic listed
in the Table of Flashback Material:
    now Beauty is passive;
    say "[reply entry][paragraph break]".
```

```
A fact is a kind of thing. A fact can be known or
unknown. A fact can be current or past.
```

```
The family circumstances is a fact. The secret identity is
a fact. The printed name of secret identity is "secret".
```

Table of Flashback Material

reply
"[if family circumstances is unknown]I wish you'd give me some
[otherwise]'I don't think you fully understand the [family
circumstances],' you say.[end if]"
"[if dramatic revelation ended in failure]'Look,' you say, trying
[otherwise]You clear your throat and allow your voice to
stray upward, into its natural register and out of this husky false
tenor you've been affecting. 'There's, er, something you should
know about my [secret identity],' you say...[end if][if dramatic
revelation ended in success] She looks impatient. 'I get it, you
know,' she says. 'I'm not stupid.'"

After printing the name of a fact (called target): now the target is current; now the target is known.

Poverty flashback is a recurring scene. Poverty flashback begins when family circumstances is current. When poverty flashback begins: strip the player; move the player to the Woodcutter's Shack.

The Woodcutter's Shack is a room. "Your family lives in a shack in the forest. There are holes in the roof, and in the winter the snow comes in -- rain, too, for that matter. The walls aren't very well-boarded, and don't keep out the wind, and even though you live in the middle of dense woods, you can never gather enough fuel to keep this place fully heated. And then there's the stench. Pigs wander freely in and out, and your three youngest brothers play on the floor."

Pigs are an animal in the shack. The pigs are scenery. The description is "They really are very grubby, dirty animals, but what's worse than that, the value of pigs has declined a lot over the last few decades. This is hard to explain to someone who has been out of touch

with the world for a while, but keeping pigs for meat is a dubious prospect when there's less and less for them to forage on." Instead of smelling the pigs: say "They smell the way animals do, when they live among their own refuse."; increase the pity of Beauty by 2.

The brothers are a man in the shack. The brothers are scenery. The description of brothers is "Hans, Franz, and Lukas. Twins and then the baby. So young, and growing up fatherless; and soon to be orphaned entirely, if your mother's health does not improve." Understand "brother" or "twin" or "twins" or "baby" or "franz" or "hans" or "lukas" as the brothers.

The untidy bed is scenery in the Shack. Mother is a woman on the untidy bed. The description of mother is "She is wasting away of a slow disease, her skin stretched tautly over bone. She hasn't been the same since your father left." On the bed is a folded letter.

The description of the letter is "Many times read over and creased, the letter explains how your father has gone away with a wealthy countess and will not return. Your mother was not able to read it herself, of course, and had to have it explained to her by the parish priest. Now she keeps it by the bed and crumples it in her fits of delirium."

Instead of kissing or touching Mother for the first time:
    say "You place a gentle kiss on her feverish brow. She looks up at you, her oldest -- yes, never mind that bit -- with a look of sincere trust and admiration.

'You'll find a way through this for us,' she says, squeezing your fingers. 'I know you will.'"; increase the pity of Beauty by 3.

Instead of kissing Mother: say "You have no more heart-rending memories of affection to recount; that one incident will have to serve, for rhetorical purposes."

Instead of waiting in the Shack: say "The wind blows sharply through the walls."

Instead of attacking someone in the Shack:
    say "Though sometimes the conditions of your life make you grouchy and impatient, you would never dream of striking a member of your own family. But from time to time you do feel the temptation."

Beauty has a number called pity. After examining something in the Woodcutter's Shack, increment the pity of Beauty. After examining mother, increase the pity of Beauty by 2. After examining the letter, increase the pity of Beauty by 3.

Poverty flashback ends when waiting or the time since poverty flashback began is five minutes.

When Poverty flashback ends:
    now family circumstances is past;
    say "...you finish describing the miserable circumstances of your home life, and allow your attention to return to the present.";
    restore the player;
    now Beauty is passive;
    if Beauty is clever and Beauty is sympathetic:
        say "'I understand,' she says slowly. 'Yes, I do. I'll do it.' She takes a deep breath and looks at you. 'We will be king together! and your family will be royalty!'";
        end the story finally;
    otherwise:
        if Beauty is sympathetic, say "'Oh dear!' she says. 'What a dreadful life!' She wrings her hands. 'No wonder you are eager to improve your lot...! But --' Her brow clears, a new thought occurring. 'You needn't marry me,

you know! We could arrange it differently! I am certain that my father would give you a large reward, instead, and then I would not be separated from my current boyfriend!'";
otherwise say "She makes a disgusted face, but she doesn't seem nearly so heart-wrung as you had hoped to make her. Tough audience, these modern princesses."

Definition: Beauty is sympathetic if the pity of Beauty is greater than 4.

To say attitude:
if Beauty is sympathetic, say "distressed on your behalf";
otherwise say "a little confused".

Saved location is a room that varies. Locker is a container. Wardrobe is a container.

To strip the player:
now every thing carried by the player is in the locker;
now every thing worn by the player is in the wardrobe;
now saved location is location.

To restore the player:
now every thing carried by the player is in the location;
now every thing in the locker is carried by the player;
now every thing in the wardrobe is worn by the player;
move the player to saved location.

Part 3 - The Other Secret

This time, we're waiting for the princess either to understand or not understand -- so we don't want to rerun the scene once it has happened successfully.

```
Beauty has a number called clue count.


Dramatic revelation is a recurring scene. Dramatic
revelation begins when attempting confidence.


To decide whether attempting confidence:
    if dramatic revelation ended in success, no;
    if secret identity is current, yes;
    no.


When dramatic revelation begins:
    strip the player;
    say "You reminisce about one of the many stops on
the way here: you had a long journey from your
homeland, and it wasn't made any easier by your
poverty, the inability to afford decent inns or plentiful
food or any kind of ride along the way.";
    move the player to the Forest Clearing;
    move the pack to the player; now the player wears
the trousers; now the player wears the shirt.


Forest Clearing is a room. "It's mid-autumn in your
memory, the pool clear and cold, gold and red leaves
floating on the surface."


The pool is scenery in the Clearing. Understand
"reflection" or "surface" or "water" as the pool. "The pool
is cold but beautiful, and the stopping place a welcome
rest." The leaves are scenery in the clearing. The
description is "Bright gold and orange and red: it's been
a sharply chilly autumn, as you have reason to know in
detail."
```

The trousers and the shirt are wearable things. The pack is a container. The pack contains ale, food, and skirt. A distraction is a kind of thing. The ale, the food, the pair of trousers, and the shirt are distractions. The description of a distraction is usually "[The item described] is not the point of this story." The shirt and the trousers are wearable. The description of the trousers is "Borrowed from your oldest brother, who is only a year younger than you. They are too long for your legs and overly snug at the hip, but no one around here pays much attention to fashion, and you're getting away with it, more or less." After examining the trousers, increment the clue count of Beauty.

Instead of examining the player during dramatic revelation:
    increment the clue count of Beauty;
    say "You cannot see yourself without reflection, but you can feel your hair loose and unbound over your shoulders."

Rule for printing the name of the skirt while taking inventory: say "one skirt you have not been able to bring yourself to part with". The description of the skirt is "Made for you by your mother, and it looks quite pretty on you. If your primary plan does not work, you may be forced to wear it again, and hope to catch a male eye... but with luck that will not be necessary." After taking inventory: increment clue count of Beauty.

Swimming is an action applying to nothing. Understand "swim" or "dive" as swimming.

Instead of swimming in the presence of the pool:
    increment clue count of Beauty;
    say "You consider going for a swim, but don't dare be caught unclad and unarmed, not here. There are too many men around, and any of them discovering you here would surely take advantage."

Instead of searching or drinking the pool:
    increment clue count of Beauty;
    say "You lean over the pool and look carefully at your reflection, your hair loose and unbound, falling around your face in waves. (That should surely give it away!)"

Instead of waiting during dramatic revelation: say "You wait for the penny to drop, for her to understand."

Dramatic revelation ends in failure when waiting or the time since dramatic revelation began is five minutes.
When dramatic revelation ends in failure:
    now secret identity is past;
    restore the player;
    now Beauty is passive;
    say "She wrinkles her nose. 'I don't understand!' she says. 'What are you trying to tell me?'

    You could weep for womankind. But you don't quite dare spell it out in so many words, not when someone might come up the stair and overhear a chance revelation."

Dramatic revelation ends in success when Beauty is clever. When dramatic revelation ends in success:
    restore the player;
    now Beauty is passive;
    say "'You're -- a girl? Like me?'

'Not much like you,' you say, glancing over her petite frame and pert nose. 'But female, at any rate.'"

Definition: Beauty is clever if the clue count of Beauty is greater than 2.

And now, since we don't really want to return to the rest of the 'marriage proposal' scene once she has learned our ID:

```
Marriage proposal ends in distraction when Dramatic
Revelation ends in success.
```

```
Compromise proposal is a scene. Compromise proposal
begins when Dramatic Revelation ends in success.
When Compromise Proposal begins: now Beauty is
passive.
```

```
Instead of asking an awake beauty about a topic listed
in the Table of Secondary Conversation when
Compromise Proposal is happening:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.
```

```
Every turn during Compromise Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Secondary Conversation:
            say "[quip entry][paragraph break]";
            if the number of filled rows in the Table of
Secondary Conversation is greater than 1, blank out the
whole row;
            make no decision.
```

```
Every turn: now Beauty is active.
```

Notice that we moved the re-activation rule down here so that the Compromise Proposal rule would fire first. There are other more complicated ways of handling order of every turn rules than by relying on text sequence alone; but we will save that for a later chapter. For now it is sufficient to depend on the order in which the rules are declared.

```
Table of Secondary Conversation
```

"'Wrong,' you say, feeling a little annoyed. 'But you see why marrying me wouldn't be an interference. You could carry on whatever romances you wished, without your father noticing.' (Probably. You'll let the pragmatic details of this work themselves out later, and hope that any children she has will look vaguely like you.)"

'Wrong,' you say, feeling a little annoyed. 'But you see why marrying me wouldn't be an interference. You could carry on whatever romances you wished, without your father even noticing.'"

"'If you can't thinking that a woman can't be the prince -- and then king -- well, there was a woman Pope, once.'

She looks at her forehead does not go away. 'But if everyone thinks you are a man... later you would be king!'

Before she can go on, you say, 'There was a woman Pope, once. Compared to that, a woman king is nothing.'"

"'So, you say, What do you think?"

[final decision]"

Her left eyebrow twitches again, which you are coming to recognize as a sign of hard mental labor. 'I think I see,' she says. [final decision]"

To say final decision:
    if Beauty is sympathetic:
        say "She considers the matter silently for some minutes, then says: 'I will do it. My beloved William will be so glad!' You imagine that William's feelings on the

```
matter will be a tad more complex than that, but do not
bother quashing her exuberance...";
        end the story finally;
    otherwise:
        say "'I still don't quite understand why you would
want this so badly as to go to all that trouble,' she
admits uneasily. Evidently you have not explained
enough to her about the poverty of your home life."



Test me with "x beauty / pour water on beauty / ask
beauty about sleep / tell beauty about poverty / smell
pigs / x mother / x letter / kiss mother / ask beauty about
marriage / tell beauty about identity / x me / look in
water / i / z / ask beauty about marriage".
```

ExampleCheese-makers *RB*

Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

As we have seen, there are a number of different ways of controlling conversation in interactive fiction, and the best choice of way will depend quite a lot on what kind of work we're writing.

One common model is to replace Inform's default ASK and TELL commands with a TALK TO command. This gives the player less control than he would otherwise have: instead of asking a character about any topic under the sun, he's restricted to seeing (or not seeing) a single sequence of text that the author has written in advance. On the other hand, such a system is harder for the player to break (since he can never ask about a topic that the author hasn't implemented), and easier for the author to tie into plot developments. If we give TALK TO different output at each scene, we get conversation that is always tied to the current state of the plot.

This is a design approach that works best in a game with a large number of short, focused scenes. For other kinds of conversation system design, compare the other examples listed in the Recipe Book.

```
"The Cheese-makers" by Phrynichus.
```

```
Chapter 1 - Replacing old talk commands and making a
new one
```

Here, using some techniques that will be discussed in the chapter on Understanding, we get rid of Inform's default handling of ASK and TELL, and create our own TALK TO action instead:

```
Understand the commands "ask" and "tell" and "say"
and "answer" as something new.


Understand "ask [text]" or "tell [text]" or "answer [text]"
or "say [text]" as a mistake ("[talk to instead]").


Instead of asking someone to try doing something:
    say "[talk to instead][paragraph break]".


Instead of answering someone that something:
    say "[talk to instead][paragraph break]".


To say talk to instead:
    say "(To communicate in [story title], TALK TO a
character.) "


Understand "talk to [someone]" as talking to.
Understand "talk to [something]" as talking to. Talking to
is an action applying to one visible thing.


Chapter 2 - Specific scenes and talking
```

Now, suppose we have a situation -- say, a stage play -- in which it is appropriate to talk to different characters at different times. During the

prologue of the play, no one else is on-stage, and the player is to address the audience directly:

```
Section 1 - Prologue


When play begins:
    now right hand status line is "416 BC";
    now left hand status line is "[location]".


Prologue is a scene. Prologue begins when play begins.


The Theater of Dionysus is a room.


The audience is a person in the Theater. "The usual
audience looks on: the priests and judges in the front
row, and then Athenians, metics, and foreigners." The
audience can be prepared or unprepared. The
description is "Have you ever seen such a company of
perjurers, pathics, and thieves?" Understand "priest" or
"priests" or "priest of dionysus" or "judge" or "judges" or
"athenians" or "metics" or "foreigners" as the audience.


Instead of talking to the player when the Prologue is
happening:
    say "There will be plenty of occasion for muttered
asides later in the play, but for now you must prepare
the audience for things to come."


Instead of talking to the audience when the Prologue is
happening:
    say "Drawing breath, you turn to the audience, and
offer them a genial, witty, colorful, and of course crude
synopsis of what they are about to see; describing all
the characters in unmistakable terms and not omitting
```

the most important of them all, your august self.";
    now the audience is prepared.

Instead of talking to the audience when the Prologue has happened:
    say "You may only direct monologues to the audience when the other actors are off-stage. Otherwise, their characters might have to notice."

Prologue ends when the audience is prepared.

But there might follow a scene in which the player shouldn't talk at all:

Section 2 - Parodos

Parodos is a scene. Parodos begins when Prologue ends.

When Parodos begins:
    move the chorus to the theater.

Instead of talking to someone during Parodos:
    say "Sssh: this moment belongs to the chorus. They've worked so hard on it, after all."

Parodos ends when the time since Parodos began is 4 minutes.

The chorus is a person. The description is "They are dressed in exaggerated rural costume and feminine masks, as they are meant to represent a company of female cheese-makers from the Spartan-occupied deme

```
of Dekeleia." Understand "cheesewives" or "cheese-
makers" or "chorus-leader" as the chorus.


Every turn during Parodos:
    repeat through Table of Choral Events:
        say "[output entry][paragraph break]";
        blank out the whole row;
        make no decision.


Table of Choral Events
output
"The chorus now begins its entry, accompanying with
anapestic song its march up the eisodos."
"The chorus draws nearer, stomping and clomping and
swinging their baskets of cheese."
"You stand aside as the chorus fills the orchestra and
dances to and fro."
"The tune of the aulos-player grows more and more
frenzied and then breaks off."
```

This last rule is a refinement borrowing from the Activities chapter, which gives characters different appearances in room descriptions depending on when we happen to look; because of the action of the play, we want to show the chorus and audience doing different things during different scenes.

```
Rule for writing a paragraph about the chorus during
Parodos:
    say "The chorus are dancing and singing their way[if
the time since Parodos began is less than 3 minutes] up
the long walkways onto the stage[otherwise] into
position in the orchestra[end if]. [The audience] appear
to be pricing their costumes to the nearest obol: woe to
the producer who cheats them of their due share of
spectacle."
```

And now a scene in which the player can talk several times to a character (Heracles) but has no useful dialogue with the chorus, the audience, or himself. The prohibition from talking to the audience after the Prologue is already written, but we'll supply some appropriate responses for talking to the player or the chorus during this scene:

Section 3 - Episode

Episode is a scene. Episode begins when Parodos ends.

When Episode begins:
    move Heracles to the theater;
    say "The chorus falls silent, which is the cue: Heracles bursts out of the scene building."

Heracles is a man. The description is "Hard to mistake in his lion skin and boots, and carrying a formidable club." Heracles wears a lion skin and boots. He carries a formidable club. Heracles can be placid or annoyed. Heracles is placid. Heracles can be satisfied, intrigued, or unsatisfied. Heracles is unsatisfied.

Instead of talking to the chorus during Episode:
    say "Your improvised flirtation with the chorus raises no response but a crude gesture from the chorus-leader, who seems to be modeling the role on Iambe."

Instead of talking to the player during Episode:
    if Heracles is annoyed:
        say "You mutter to yourself about men with more appetite than brain. The actor playing Heracles ignores you, but it's good odds he's scowling under his mask. He hates it when anyone but himself ad-libs for attention.";
    otherwise:
        now Heracles is annoyed;
        say "'By the dog, he'll eat me if he gets a chance,' you mutter aside. [paragraph break]'What's that you say, my ignoble friend?' demands Heracles, hefting his club. He's not entirely joking: you've left the script just now."

Instead of talking to Heracles when Heracles is unsatisfied during Episode:
    say "'Dear Heracles, friendly Heracles,' you begin, cringing out of the way as he responds with one of his affectionate ox-killing punches to the shoulder. [paragraph break]But Heracles falls still, and looks almost thoughtful, as tell him you know how he may rout the Spartans, woo all twenty-four lactic ladies, and tame his savage gut with a bathtubful of porridge. [paragraph break]'Speak on, little man,' he says.";
    now Heracles is intrigued.

Instead of talking to Heracles when Heracles is intrigued during Episode:
    say "It takes several exchanges for him to wrap his one-inch brain around your ten-inch plan; but in the end he embraces the scheme, the women, and your humble self.";
    now Heracles is satisfied.

Every turn when not talking to someone during Episode:
    repeat through Table of Episodic Events:
        say "[output entry][paragraph break]";
        blank out the whole row;
        make no decision.

Table of Episodic Events
output
"With a fart and a roar, Heracles asks the world at large, and you in particular, where his dinner might be."
"In epic diction, Heracles invites the dairy-mistresses, whey-matrons, and concubines of curd to supply him a supper from their ample baskets."
"Heracles and the chorus banter about the proclivities of cheese-wives. The chorus suggest that Heracles, as a son of Zeus, must know something about the appetites of which they speak."
"Heracles boasts that a man like himself can perform any feat, but only when his belly is full. Coyly, the matrons prance and dance, skip and gambol out of his grasp, singing mockingly about heads of garlic and loaves of sesame-crusted bread."
"The song of the feta fanciers now turns to pots of honey and new-made wine, borrowing verses from last

year's Lenaia winner, 'The Bees'. With a jolt, you realize that you've missed your cue and the chorus are filling in for you."
"Playing for time, the chorus-leader elaborates a whole banquet: rabbit stew, shanks of lamb, spitted quails, eels from lake Copais. Heracles looks as near swooning as any girl fresh from Brauron."
"The chorus-leader extends the list of delicacies to include ox-brains, ham-hocks, barley, mullet, carrots, pigeons, lentils, radishes, peas, and apples both wine-dark and golden. The audience shifts on the benches. An expression of gloom settles over the Priest of Dionysus in the front row."
"Inspired by Euripides['] own Muse, the chorus-leader invents a mock-Alcaean hymn on the merits of chervil. This is clearly his swan-song: if you don't speak at last, the play will come to a halt."
"Silence descends."

Rule for writing a paragraph about Heracles during Episode:
    say "[Heracles] stands at the center of the orchestra, with members of [the chorus] ranged on either side. [paragraph break][The audience] appear to be reserving their judgement, though they show signs of restiveness at the usual jokes: must there be a Heracles in [italic type]every[roman type] play?"

Episode ends successfully when Heracles is satisfied.

When Episode ends successfully:
    say "That, of course, is your cue: you're to come back on as Pan thirty verses from now, and it takes time to put on the hooves and the woolly-legged trousers.";
    end the story saying "You exit".

Episode ends disastrously when the number of filled rows in the Table of Episodic Events is 0.

When Episode ends disastrously:
    end the story saying "The production has crashed to a halt".

Test me with "ask audience about me / tell audience about me / audience, hello / audience, jump / talk to me / talk to audience / g / talk to chorus / look / x heracles / talk to me / talk to audience / z / look / talk to heracles / g".

Chapter 11: Phrases

; *§11.9. While*; *§11.10. Repeat*; *§11.11. Repeat running through*; *§11.12. Next and break*; *§11.13. Stop*

Contents of *Writing with Inform*
Chapter 10: Scenes
Chapter 12: Advanced Actions
Indexes of the examples

§11.1. What are phrases?

Phrases are instructions to Inform to do something, or to decide whether something is true or false, or to produce a value, or to say something. Inform has around 350 phrases built-in, and the chapters so far have already defined about 100 of those. In this chapter we'll see some key phrases for organising instructions of what to do, and also see how to define entirely new phrases.

Just to run through the four sorts of phrase with examples:

(a) Phrases to do something. These are the ones used in the body of a rule. For example,

```
When Train Stop begins:
    move the Flying Scotsman to the Station;
    say "The Flying Scotsman pulls up at the platform."
```

Rules like this begin with a "preamble", the beginning part which tells Inform when or how they apply, and then follow on with a list of instructions - here, just two of them. "move ... to ..." and "say ..." are both phrases. Inform provides about 130 of these built-in. It's actually not quite true that they all do something, because one of them is:

do nothing

This phrase does nothing at all. It is very occasionally useful to make a rule which does nothing:

```
This is the largely ineffective rule:
    do nothing.
```

(b) Phrases to decide whether a condition is true. These are the ones which can be used in an "if":

```
if action requires light: ...
```

Not all conditions come from phrases. For example, "if the front door is closed" and "if Peter is wearing the sandals" have meanings which come from the verbs "to be" and "to wear". Inform provides about 60 built-in conditions, which give a friendly wording for questions which would be lengthy or difficult to write in any other way.

(c) Phrases to decide a value. For example:

```
square root of 16
```

produces a number, 4 of course, and can be used whenever a number is expected. Inform provides about 100 built-in phrases like this.

(d) Text substitutions. These are actually just phrases whose definition begins with "To say ...". Example:

```
"It's now [time of day in words]."
```

Inform provides about 60 built-in text substitutions.

Start of Chapter 11: Phrases

Back to Chapter 10: Scenes: §10.9. Why are scenes designed this way?

Onward to §11.2. The phrasebook

§11.2. The phrasebook

The Phrasebook is Inform's collection of recognised phrases, and it can always be browsed using the Index panel of the same name. Even the smallest project has a good-sized phrasebook, since it contains all of the built-in phrases. But most projects

also define new phrases of their own.

Here is a simple definition of a new phrase:

```
To spring the trap:
    say "'Sproing!' go the hinges and, with a flash of silver, the
enormous blades whisk together!";
    end the story.
```

Inform allows us to use whatever conventions of layout we prefer, but it's customary to use indentation like this, dividing off the preamble from the phrases which follow. As can be seen, definitions of new phrases look very like rules.

What makes this definition a simple one is that the wording is fixed. The only way to use this would be from another phrase or rule, like so:

```
Instead of entering the cage:
    spring the trap.
```

In the next section we'll see how to give more complicated definitions which, like "move ... to ...", allow for the wording to change with the circumstances.

Start of Chapter 11: Phrases
Back to §11.1. What are phrases?
Onward to §11.3. Pattern matching

§11.3. Pattern matching

In this section, let's make the following new phrase:

```
To admire (item - an object):
    say "You take a long look at [item].".
```

This does very little, of course, but it does allow the wording to be different each time the phrase is used:

```
admire the diamonds;
admire Mr Cogito;
admire the honey sandwich;
```

and our single definition covers all of these possibilities. The bracketed part of the definition, "(item - an object)", tells Inform to expect an object in that position, and Inform enforces this carefully. So this definition might tell Inform what "admire the barricade" means, but not what

```
admire "blue cheese";
admire 63;
```

mean. Unless some other definition sorts the matter out, Inform will reply to uses like this with a Problem message:

```
Problem. You wrote 'admire 63', but '63' has the wrong kind of
value: a number rather than an object.
```

The object does not need to be named literally, but can be anything which works out to be an object: for instance,

```
After dropping something in the Auction House:
    admire the noun.
```

which Inform allows because "noun", here, is a name for the object which is being acted on.

Inform decides which definition to apply in a process called "pattern matching".

The bracketed part of the example definition has the form "(name - kind)". The definition only applies if the text supplied agrees with the "kind" part - for instance, the diamonds agreed with "object", but 63 did not. If the definition does apply, then the Inform works through the rest of the phrase using "name" to mean whatever value matched. For example:

```
To slam shut (box - an open container):
    say "With great panache, you slam shut [the box].";
    now the box is closed.
```

When this phrase is followed, "box" means whatever open container the pattern-matcher found when it was called for. For example, if Inform reads

```
slam shut the Dutch armoire;
```

then it acts on this by following the definition of "slam shut ...", using the Dutch armoire object as the value of "box", so it prints:

```
With great panache, you slam shut the Dutch armoire.
```

and renders it closed.

In fact any description can be given in the definition, and that includes a single, specific value. For instance, we could define:

```
To grant (bonus - a number) points:
    increase the score by the bonus.
```

```
To grant (bonus - 7) points:
    say "You shiver uncontrollably."
```

which would withhold this unlucky bounty. That would mean that:

```
grant 7 points;
grant seven points;
```

would each produce uncontrollable shivers, because Inform uses the definition applying to the number 7; but

```
grant six points;
```

would increase the score by 6. In general Inform always follows the principle that more specific definitions take priority over more general ones. So although the definitions:

```
To grant (bonus - a number) points: ...
To grant (bonus - 7) points: ...
```

both apply to the case of "grant 7 points", Inform uses the second, because it's the more specific of the two possibilities.

Sometimes it will not be possible to tell if the value supplied meets the requirements until the story is actually playing. If, at run-time, no definition fits some phrase which has to be carried out, a run-time problem message is produced.

Finally, and more straightforwardly, we can specify variations in wording using slashes between alternative words in a "To ..." definition. For instance:

```
To grant (bonus - a number) point/points: ...
```

allows the final word to be either "point" or "points". Slashes like this can only be used with literal words, not bracketed values, and give alternative forms only of a single word at a time; the alternative "--" means "no word at all", and thus makes it optional:

```
To grant (bonus - a number) point/points/--: ...
```

makes "grant 3" do the same as "grant 3 points".

If we need more variation than that, we should make more than one definition.

Start of Chapter 11: Phrases

Back to §11.2. The phrasebook

Onward to §11.4. The showme phrase

Example 169: ★ **Ahem** Writing a phrase, with several variant forms, whose function is to follow a rule several times.

Example 170: ★★ **Ferragamo Again** Using the same phrase to produce different results with different characters.

§11.4. The showme phrase

We've already seen the SHOWME command, which can be typed into the Story

panel to look at the state of something, usually a thing or room. SHOWME is a testing command which has no effect once the work is released; eventual players can't use it.

Inform also has a phrase called "showme", which works in much the same way:

> **showme** (value)
>
> This phrase is intended for testing purposes only. If used in a story file running inside the Inform application, it prints a line of text showing the given value and its kind; in a Released story file, it does nothing at all. Example:
>
> ```
> When play begins: showme 11.
> ```
>
> produces
>
> ```
> number: 11
> ```
>
> More usefully:
>
> ```
> Every turn: showme the score.
> ```
>
> Now, every turn, we get a line in the story's transcript like so:
>
> ```
> "score" = number: 0
> ```
>
> Inform uses the quotation marks and equals sign to show that it had to do some work to find the answer. "score" wasn't a constant value - it was a variable, and Inform had to look up the current value.

"showme" is a convenient way to see what's going on inside a phrase which isn't behaving as expected, or to find out the kind of a value. Here are some trickier examples. Suppose our design includes:

```
The matching key of the blue door is the brass Yale key.
```

If we then try this:

```
When play begins:
    showme matching key of the blue door.
```

we get, when the story starts up,

```
"matching key of the blue door" = object: brass Yale key
```

Why is this an "object", when we know that the key is actually a "thing"? After all, if we "showme key" instead, we get:

```
thing: brass Yale key
```

The answer is a little technical: it's because Inform guarantees that the matching key is always an object, but not that it's always a thing - it just happens to be a thing at the moment. There's not really a contradiction, because a "thing" is a kind of "object", so in fact the key is both. If we try "showme matching key", we get something like this:

```
objects valued property: property 23
```

which is even more technical - people never need to print the names of abstract property names during play, so Inform doesn't provide any good way of doing it. It is reduced to printing out an internal ID number ("property 23") instead of the name ("matching key"). This can't be helped: "showme" is a way to lift the lid and see what's going on inside Inform's machinery, and some of the corners are dark.

All the same, "showme" can be very useful in tinkering with rules to make them work properly. It prints nothing at all in a Release version of a project, so it's impossible for these private notes to be shown accidentally to our eventual readers.

Start of Chapter 11: Phrases

Back to §11.3. Pattern matching

Onward to §11.5. Conditions and questions

§11.5. Conditions and questions

A variety of "conditions" have already appeared in this documentation. A condition is a phrase which describes a situation which might be true, or might be false, and examples might include:

```
Mr Kite is in Bishopsgate
the score is greater than 10
Sherlock Holmes suspects a woman
```

These are all examples of sentences, formed by putting nouns either side of a verb, and clearly a wide range of conditions can be written this way. But there are also a few special conditions built into Inform which have a fixed wording, and test questions difficult to address with ordinary sentences. For instance:

if in darkness:

This condition is true if the player currently has no light to see by. Note that the test is more complicated than simply testing

```
if the player is in a dark room, ...
```

since the player might have a torch, or be inside a cage which is itself in a dark room, and so on.

Another example of a condition not easily written as a sentence is:

if player consents:

This condition is unusual in doing something and not simply making a silent check: it waits for the player to type YES (or Y) or NO (or N) at the keyboard, and then is true if the answer was yes. Example:

```
say "Are you quite sure you want to kiss the Queen? ";
if the player consents:
    ...
```

Whether it's put to the player like this or not, testing a condition is really asking a question, and there is always a yes/no answer. In Inform this answer is not usually a value (unlike in some other computer programming languages), but it can be made into one.

Firstly, we need a special kind of value to hold answers like this. It's called "truth state", and it has just two possible values, written as "true" and "false". We then need:

whether or not (a condition) **... truth state**

This phrase converts a condition into its result as a value, which is always either "true" or "false". Example:

```
whether or not 20 is an odd number
```

produces the truth state "false". This is mostly useful for storing up results to look at later:

```
let victory be whether or not all the treasures are in the
cabinet;
```

and then subsequently:

```
if victory is true, ...
```

As another example, in most stories this:

```
When play begins:
    showme whether or not in darkness.
```

...will produce a line:

```
"whether or not in darkness" = truth state: false
```

In short, "truth state" is a kind of value like any other. That means it can be the kind of a variable:

```
Salvation earned is a truth state that varies.
```

and it can similarly be used in table columns, lists, or anywhere else where values are allowed.

Start of Chapter 11: Phrases

Back to §11.4. The showme phrase

Onward to §11.6. If

Example 171: ★★ **Proposal** Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

§11.6. If

Inform's most powerful phrases are those which control the others, making them repeat, or be skipped.

if (a condition) **,** (a phrase)

or:
if (a condition)**:**

This phrase causes the single phrase, or block of phrases, following it to be obeyed only if the condition is true. (If the condition must contain a comma for some reason, the block form should be used.) Example:

```
if the red door is open, say "You could try going east?"
```

The sense of an "if" can be reversed by using the word "unless" instead:

unless (a condition) **,** (a phrase)

or:
unless (a condition)**:**

This phrase causes the single phrase, or block of phrases, following it to be obeyed only if the condition is false. (If the condition must contain a comma for some reason, the block form should be used.) Example:

```
unless the red door is closed, say "You could try going
east?"
```

"Unless" is clearly unnecessary, but it can be a good way to make the source text easier for humans to read.

As we have seen, there are many different forms of condition in Inform. They usually take a form quite like an assertion sentence, except that they're questions and not statements of fact. For example:

```
if the score is 10, ...
if all of the people are in the Atrium, ...
```

Questions like this are checked by Inform to see if they make sense. The following doesn't, for instance:

```
if 10 is a door, say "Huzzah!";
```

This produces the baffled reply:

```
Problem. In the line 'if 10 is a door, say "Huzzah!"' , I can't
determine whether or not '10 is a door', because it seems to ask if a
number is some sort of door.
```

Start of Chapter 11: Phrases

Back to §11.5. Conditions and questions

Onward to §11.7. Begin and end

§11.7. Begin and end

In practice it is not enough to apply "if" to a single phrase alone: we want to give a whole list of phrases to be followed repeatedly, or to be followed only if a condition holds.

We do this by grouping them together, and there are two ways to do this. One is as follows:

```
To comment upon (whatever - a thing):
    if whatever is transparent, say "I see right through this!";
    if whatever is an open door:
        say "Oh look, an open door!";
        if whatever is openable, say "But you could always shut it."
```

Here we group two phrases together under the same "if". Note that the comma has been replaced by a colon, and that the indentation in the list of phrases shows how they are grouped together. In the example above, the source moves two tabs in from the margin; the maximum allowed is 25.

Indentation is the convention used in this manual and in the examples, but not everybody likes this Pythonesque syntax. So Inform also recognises a more explicit form, in which the beginning and ending are marked with the words "begin" and "end":

```
To comment upon (whatever - a thing):
    if whatever is transparent, say "I see right through this!";
    if whatever is an open door
    begin;
        say "Oh look, an open door!";
        if whatever is openable, say "But you could always shut it.";
    end if.
```

(Pythonesque because it's a style popularised by the programming language Python, named in turn after "Monty Python's Flying Circus".)

Start of Chapter 11: Phrases

Back to §11.6. If

Onward to §11.8. Otherwise

Example 172: **Matreshka** A SEARCH [room] action that will

open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

Example 173: **Princess and the Pea** The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

---

§11.8. Otherwise

We often need code which does one thing in one circumstance, and another the rest of the time. We could do this like so:

```
if N is 2:
    ...
if N is not 2:
    ...
```

but this is not very elegant, and besides, what if the action we take when N is 2 changes N so that it becomes something else?

Instead we use "otherwise":

otherwise if (a condition)

or:
otherwise unless (a condition)

or:
otherwise (a phrase)

or:
else if (a condition)

or:

else unless (a condition)

or:
else (a phrase)

This phrase can only be used as part of an "if ...:" or "unless: ...", and provides an alternative block of phrases to follow if the first block isn't followed. Example:

```
if N is 2:
    ...
otherwise:
    ...
```

When there is only a single phrase we can use the shortened form:

```
if N is 2, say "Hooray, N is 2!";
otherwise say "Boo, N is not 2...";
```

We can also supply an alternative condition:

```
if N is 1:
    ...
otherwise if N is 2:
    ...
otherwise if N is greater than 4:
    ...
```

At most one of the "..." clauses is ever reached - the first which works out.

If the chain of conditions being tried consists of checking the same value over and over, we can use a convenient abbreviated form:

if (value) **is:**

This phrase switches between a variety of possible blocks of phrases to follow, depending on the value given. Example:

```
if the dangerous item is:
    -- the electric hairbrush:
        say "Mind your head.";
    -- the silver spoon:
        say "Steer clear of the cutlery drawer."
```

One alternative is allowed to be "otherwise", which is used only if none of the other cases apply, and which therefore guarantees that in any situation exactly one of the blocks will be followed.

```
if N is:
    -- 1: say "1.";
    -- 2: say "2.";
    -- otherwise: say "Neither 1 nor 2.";
```

This form of "if" layout is not allowed to use "begin" and "end" instead of indentation: it would look too messy, and would scarcely be an abbreviation. It is also not allowed to use "unless" instead of "if", because the result would be too tangled to follow.

Start of Chapter 11: Phrases

Back to §11.7. Begin and end

Onward to §11.9. While

Example 174: **Numberless**
A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

§11.9. While

The next control phrase is "while", which has the form:

while (a condition)**:**

This phrase causes the block of phrases following it to be repeated over and over for as long the condition is true. If it isn't even true the first time, the block is skipped over and nothing happens. Example:

```
while someone (called the victim) is in the Crypt:
    say "A bolt of lightning strikes [the victim]!";
    now the victim is in the Afterlife;
```

We must be careful not to commit mistakes like the following:

```
while eggs is eggs:
    say "again and ";
```

which, as sure as eggs is eggs (which is very sure indeed), writes out

```
again and again and again and again and again and ...
```

forever. (Inform won't prevent this: we will find out the hard way when the story is played.) While we would probably never write anything so blatant as that, the mistake is all too easy to commit in disguised form. We should never design a loop, as repetitions like this are called, without worrying about if and when it will finish.

As with "if", we can use "begin" and "end" instead of a tabulated layout if we want to --

```
while ...
begin;
    ...
end while.
```

(The "begin" of an "if" must of course match an "end if", not an "end while", and so on.)

Experience shows that it is much more legible to lay out "while" loops as blocks, even in these rare cases when only a single phrase forms the body of the block.

Start of Chapter 11: Phrases

Back to §11.8. Otherwise

Onward to §11.10. Repeat

§11.10. Repeat

The other kind of loop in Inform is "repeat". The trouble with "while" is that it's not obvious at a glance when or whether the loop will finish, and nor is there any book-keeping to measure progress. A "repeat" loop is much more predictable, and is more or less certain to finish.

There are several forms of "repeat", of which the simplest is similar to the old FOR/NEXT loop from the home-computer programming language BASIC, for those with long memories:

repeat with (a name not so far used) **running from** (arithmetic value) **to** (arithmetic value)

or:
repeat with (a name not so far used) **running from** (enumerated value) **to** (enumerated value)**:**

This phrase causes the block of phrases following it to be repeated once for each value in the given range, storing that value in the named variable. (The variable exists only temporarily, within the repetition.) Example:

```
repeat with counter running from 1 to 10:
    ...
```

This, and runs through the given phrases ten times. Within those phrases, a special value called "counter" has the value 1 the first time through, then the value 2, then 3 and so on up to 10. (It can of course be called whatever we like: this is only an example.) The range can be from any kind where ranges make sense - anything on which arithmetic can be done, so for instance

```
repeat with moment running from 4 PM to 4:07 PM:
    ...
```

and also any enumeration:

```
Colour is a kind of value. The colours are red, orange,
yellow, green, blue, indigo and violet.


...
```

```
repeat with hue running from orange to indigo:
    ...
```

We are allowed to "nest" loops, that is, to put one inside another.

```
To plot a grid with size (S - a number):
    repeat with x running from 1 to S:
        say "Row [x]:";
        repeat with y running from 1 to S:
            say " [y]";
        say "."
```

If we then write

```
plot a grid with size 5;
```

then the result is

```
Row 1: 1 2 3 4 5.
Row 2: 1 2 3 4 5.
Row 3: 1 2 3 4 5.
Row 4: 1 2 3 4 5.
Row 5: 1 2 3 4 5.
```

Thus the innermost phrase, the say which mentions "y", happens 25 times.

Whenever dealing with numbers in Inform we may need to remember that if the Settings for the project are set to use the Z-machine, the range is restricted to -32768 up to 32767. Repeating with a counter up to exactly 32767 is hazardous, because the counter can never break through this barrier: it's infinity, so far as Inform is concerned, and that can cause the repetitions to go on forever. (On Glulx, numbers can be very much larger.)

Start of Chapter 11: Phrases

Back to §11.9. While

Onward to §11.11. Repeat running through

Example 175: ★ **Wonka's Revenge** A lottery drum which

redistributes the tickets inside whenever the player spins it.

§11.11. Repeat running through

Inform is not used very much for numerical work, so the kind of repeat loop described in the previous section is not much used. Inform's natural domain is really the world of things and rooms, so the following kind of repeat is much more useful.

repeat with (a name not so far used) **running through** (description of values)**:**

This phrase causes the block of phrases following it to be repeated once for each value matching the description, storing that value in the named variable. (The variable exists only temporarily, within the repetition.) Example:

```
repeat with item running through open containers:
    ...
```

If there are no containers, or they are all closed, the phrases will not be followed at all. Inform will issue a Problem message if the range of the loop may be infinite: for example, it won't allow:

```
repeat with X running through odd numbers:
    ...
```

On the other hand it will allow:

```
repeat with T running through times:
    ...
```

which repeats 1440 times, starting with T at midnight and finishing at 11:59 PM. See the Kinds index for which kinds of value can be repeated through.

As with counting the "number of ..." objects satisfying some property, we can run through a wide variety of possibilities - any description whose range is possible for Inform to search. For example:

```
repeat with dinner guest running through the people in the Dining
Room:
    ...

repeat with possession running through things carried:
    ...

repeat with event running through non-recurring scenes which are
happening:
    ...
```

The following lists the whereabouts of all men in lighted rooms:

```
repeat with suspect running through the men who are in a lighted
room:
    say "[The suspect] is in [the location of the suspect].";
```

One small note of caution: if what the "repeat" loop does is to change the things being repeated through, changing in particular whether items not yet reached will qualify to be repeated through, the results can be unexpected. Rather than writing "repeat with X running through D", it may be safer to try "while there is D (called X)", though note that this will only finish if X is always changed so that it no longer qualifies.

Start of Chapter 11: Phrases

Back to §11.10. Repeat

Onward to §11.12. Next and break

Example 176: ★★ **Strictly Ballroom** People who select partners for dance lessons each turn.

---

§11.12. Next and break

So "repeat" and "while" phrases cause a block of other phrases to be repeated, over

and over. The number of repetitions and the flow of "control" has so far been controlled only by the way the original loop was described.

But in fact it's also possible to change this from inside the block being repeated, using these:

next

This phrase can only be used inside a "repeat" or "while" block, and causes the current repetition of the block to finish immediately. That either means the next repetition begins, or (if we are already at the last one) the loop ends too. Example:

```
repeat with X running from 1 to 10:
    if X is 4, next;
    say "[X] ".
```

produces the text "1 2 3 5 6 7 8 9 10 ", with no "4" because the "say" phrase was never reached on the fourth repetition.

In Monopoly terms, "next" is "Advance to Go" rather than "go directly, do not pass Go, do not collect $200" - the next iteration begins with the variable, if there is one, having cleanly moved on to the next value, just as if the loop had been run through in the normal way. ("Next" is called "continue" in a fair number of programming languages, so Inform issues a specific problem message to help people who forget this.)

break

This phrase can only be used inside "repeat", "while" block, and causes both the current repetition and the entire loop to finish immediately. Example:

```
repeat with X running from 1 to 10:
    if X is 7, break;
    say "[X] ".
```

produces the text "1 2 3 4 5 6 ", with nothing after "6" because the loop was broken at that point. The "say" wasn't reached on the 7th repetition, and the 8th, 9th and 10th never happened.

Start of Chapter 11: Phrases
Back to §11.11. Repeat running through
Onward to §11.13. Stop

§11.13. Stop

Now that it's possible to define phrases where different things are done in different circumstances, we sometimes want to halt early. This is what "stop" is for.

stop

This phrase causes the current rule to end immediately. It is most often used in the definition of other phrases:

```
To judge the score:
    if the score is 0, stop;
    say "The score is [score in words] more than it was a half-
hour ago."
```

In the case when the score is 0, the "stop" ends the phrase immediately, so that the subsequent text is printed only if the score is not 0.

"Stop" can also be used in action rules, though this is not very good style - it's clearer to use "stop the action", which is exactly equivalent.

Start of Chapter 11: Phrases
Back to §11.12. Next and break
Onward to §11.14. Phrase options

§11.14. Phrase options

There are sometimes several slightly different ways to perform a given task but which have substantially the same definition. In the following example:

```
To go hiking, into the woods or up the mountain:
    if into the woods, say "Watch out for badgers.";
    if up the mountain, say "Better take your compass.";
    say "You go hiking."
```

...a phrase has been set up which can be used in three ways:

```
go hiking;
go hiking, into the woods;
go hiking, up the mountain;
```

Note that commas must be used to divide these "phrase options" from the rest of the text of the phrase. Within the definition of the phrase, the option's name is a valid condition, and

```
if up the mountain, ...
```

tests whether it is set; we can also test if it is not set using:

```
if not up the mountain, ...
```

A more substantial example from the Standard Rules is given by a phrase used mostly for internal, technical reasons:

list the contents of (object)

This phrase produces a list of all things whose holder is the given object, according to Inform's traditional conventions for room descriptions and inventory listings. Example:

```
list the contents of Marley Wood, as a sentence, with
newlines
and including all contents;
```

Where this is possible, it's generally better to use "[list of things in ...]" instead, which produces the same result in an acceptable way for the middle of a sentence.

Note that this phrase is allowed to have multiple options specified, whereas "go hiking" above was not: this is because it was defined thus:

> To list the contents of (something - an object), with newlines, indented, as a sentence, including contents, including all contents, giving inventory information, giving brief inventory information, using the definite article, listing marked items only, prefacing with is/are, not listing concealed items, suppressing all articles and/or with extra indentation: ...

The significant difference is the word "and/or" instead of "or", which signals that more than one option can apply at a time.

Start of Chapter 11: Phrases

Back to §11.13. Stop

Onward to §11.15. Let and temporary variables

Example 177: ★★ **Equipment List** Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

---

§11.15. Let and temporary variables

A variable, as we have seen, is a name for a value which changes, though always remaining of the same kind. For instance, if "target" is a number variable (or "number that varies") then it may change value from 2 to 4, but not from 2 to "fishknife".

To make complicated decisions, phrases often need to remember values on a temporary basis. We have already seen this for the counter in a "repeat" loop, which exists only inside that loop, and then is no longer needed.

We can also make temporary variables using "let":

let (a name not so far used) **be** (value)

or:

let (a temporary named value) **be** (value)

This phrase creates a new temporary variable, starting it with the value supplied. The variable lasts only for the present block of phrases, which certainly means that it lasts only for the current rule. Examples:

```
let outer bull be 25;
let the current appearance be "reddish brown";
let the special room be Marley Wood;
```

The kinds of these are deduced from the values given, so that, for instance,

```
say "The outer bull scores [the outer bull in words] when you
practice archery in [special room]."
```

produces

```
The outer bull scores twenty-five when you practice archery
in Marley Wood.
```

The variable name should be a new one; if it's the name of an existing one, then the kinds must agree. So:

```
let outer bull be 25;
let outer bull be 50;
```

is a legal combination, because the second "let" simply changes the value of the existing "outer bull" variable to a different number.

let (a name not so far used) **be** (name of kind)

This phrase creates a new temporary variable of the given kind. The variable lasts only for the present block of phrases, which certainly means that it lasts only for the current rule. Example:

```
let inner bull be a number;
```

The variable created holding the default value for that kind - in this case, the number 0. A handful of very obscure kinds have no default values, and then a problem message is produced. Inform also disallows:

```
let the conveyance be a vehicle;
```

because temporary variables aren't allowed to have kinds more specific than "object". (This is a good thing: suppose there are no vehicles in the world?) It's quite safe in such cases to use

```
let the conveyance be an object;
```

instead, which creates it as the special object value "nothing".

Temporary variables made by "let" are only temporarily in existence while a phrase is being carried out. Their values often change: we could say

```
let x be 10;
now x is 11;
```

for instance, or indeed we could "let x be 10" and then "let x be 11". But although we are allowed to change the value, we are not allowed to change the kind of value. The name "x" must always have the same kind of value throughout the phrase to which it belongs, so the following will not be allowed:

```
let x be 45;
now x is "Norway";
```

(The difference between using "let" and "now" here is that "let" can create a new temporary variable, whereas "now" can only alter things already existing: on the other hand, "now" can change many other things as well, whereas "let" applies only to temporary variables.)

Start of Chapter 11: Phrases
Back to §11.14. Phrase options
Onward to §11.16. New conditions, new adjectives

Example 178: ★ **M. Melmoth's Duel** Three basic ways to inject random or not-so-random variations into text.

§11.16. New conditions, new adjectives

We can create new conditions by defining a phrase with "to decide whether" (or equivalently "to decide if"):

```
To decide whether danger lurks:
    if in darkness, decide yes;
    if the Control Room has been visited, decide no;
    decide yes.
```

If the player is indeed in darkness, the decision is "yes" because the "decide yes" stops the process right there. We can now write, for instance,

```
if danger lurks, ...
```

In fact, "danger lurks" is now a condition as good as any other, and can be used wherever a condition would be given. Rules can apply only "when danger lurks", for instance.

yes

or:
decide yes

This phrase can only be used in the definition of a phrase to decide whether a condition holds. It ends the decision process immediately and makes the condition true.

no

or:

decide no

This phrase can only be used in the definition of a phrase to decide whether a condition holds. It ends the decision process immediately and makes the condition false.

We can also supply definitions of adjectives like this. So far, new adjectives have been defined like so:

```
Definition: a supporter is occupied if it is described and something is
on it.
```

If we want to give a definition which involves more complex logic, we can use a special form allowing us to make arbitrary decisions. In this longer format, the same definition would look like so:

```
Definition: a supporter is occupied:
    if it is undescribed, decide no;
    if something is on it, decide yes;
    decide no.
```

Here "it" refers to the supporter in question. Note that there are now two colons in this sentence, one after "Definition", the other after the clause being defined. But that apart, it's a phrase like any other: it must end in "yes" or "no" just as the "danger lurks" example must. "Decide no" and "decide yes" are needed so often that they can be abbreviated by leaving out "decide":

```
Definition: a supporter is occupied:
    if it is undescribed, no;
    if something is on it, yes;
    no.
```

Start of Chapter 11: Phrases

Back to §11.15. Let and temporary variables

Onward to §11.17. Phrases to decide other things

Example 179: ★★★ **Owen's Law** OUT always means "move

to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

§11.17. Phrases to decide other things

A condition is a yes/no decision, but we can also take decisions where the result is a value. Suppose we want to create a concept of the "grand prize", which will have different values at different times in play. Each time the "grand prize" is referred to, Inform will have to decide what its value is, and the following tells Inform how to make that decision:

```
To decide which treasure is the grand prize:
    if the Dark Room has been visited, decide on the silver bars;
    decide on the plover's egg.
```

Note that we have to say what kind the answer will be: here it's a kind of thing called "treasure" (which we're supposing has already been created), and as it turns out only two treasures are ever eligible anyway (we're also supposing that the plover's egg and the silver bars are treasures already created, of course). And note also that the phrase must in all cases end with a "decide on ..." to say what the answer is:

decide on (value)

This phrase can only be used in the body of a definition of a phrase to decide a value. It causes the calculation to end immediately, with the outcome being the given value, which must be of the kind expected. Example:

```
To decide which number is double (N - a number):
    let D be N times N;
    decide on D.
```

Now that we have "grand prize" created, we can use it just as we would use any other value, so for instance:

```
if taking the grand prize, ...
```

As this is something of a dialect difference between English speakers, "what" and "which" are synonymous here, i.e., we could equally well write something like:

```
To decide what number is the target score: ...
```

(A phrase to decide if something-or-other is exactly the same thing as a phrase to decide a truth state, and indeed, if we want to then we can use "decide on T", where T is a truth state, in its definition. For instance:

```
To decide if time is short:
    if the time of day is after 10 PM, decide on true;
    ...
    decide on whether or not Jennifer is hurried.
```

"Decide on true" is exactly equivalent to the more normally used "decide yes", and of course it is optional. The last line is more interesting since it effectively delegates the answer to another condition.)

Start of Chapter 11: Phrases

Back to §11.16. New conditions, new adjectives

Onward to §11.18. The value after and the value before

Example 180: ★ **Witnessed 2** A piece of ghost-hunting equipment that responds depending on whether or not the meter is on and a ghost is visible or touchable from the current location.

Example 181: ★★★ **A Haughty Spirit** Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

---

§11.18. The value after and the value before

A point which has come up several times in recent chapters is that enumerated kinds of value have a natural ordering. For example, if we write:

```
Colour is a kind of value. The colours are red, orange, yellow, green,
blue, indigo and violet.
```

...then we not only have seven possible values, we have put them into a sequence, in order of their naming. We can't perform arithmetic on colours, of course, but we can perform comparisons on them. Thus "red < yellow" is true, while "green >= violet" is not. (More on comparisons in the chapter on Numbers and Equations, which also covers arithmetic.)

It's also sometimes useful to get at the sequence directly. First, the two ends:

first value of (name of kind) **... value**

This phrase produces the first-created value of the given kind, which should be an enumeration. Example: if we have

```
Colour is a kind of value. The colours are red, orange,
yellow, green, blue, indigo and violet.
```

then "first value of colour" is red.

last value of (name of kind) **... value**

This phrase produces the last-created value of the given kind, which should be an enumeration. Example: if we have

```
Colour is a kind of value. The colours are red, orange,
yellow, green, blue, indigo and violet.
```

then "last value of colour" is violet.

And now how to step forward and back:

(name of kind) **after** (enumerated value) **... value**

This phrase produces the next-created value of the given kind, which should be an enumeration. Example: if we have

```
Colour is a kind of value. The colours are red, orange,
yellow, green, blue, indigo and violet.
```

then "colour after orange" is yellow.

(name of kind) **before** (enumerated value) **... value**

This phrase produces the previous-created value of the given kind, which should be an enumeration. Example: if we have

```
Colour is a kind of value. The colours are red, orange,
yellow, green, blue, indigo and violet.
```

then "colour before blue" is green.

Start of Chapter 11: Phrases

Back to §11.17. Phrases to decide other things

Onward to Chapter 12: Advanced Actions: §12.1. A recap of actions

Example 182: ★ **Entropy** All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

Example 183: ★★★ **The Hang of Thursdays** Turns take a quarter day each, and the game rotates through the days of the

week.

Examples from Chapter 11: Phrases

Start of this chapter
Chapter 12: Advanced Actions
Indexes of the examples

169 **ExampleAhem** *RB*

Writing a phrase, with several variant forms, whose function is to follow a rule several times.

As we see in the example here, it is possible to use slashed variations in more than one place in a phrase, and to offer a number of separate forms. The main rule of thumb to remember is that value inputs for the phrase should always be separated by some text; so

```
To do/follow (chosen rule - a rule) exactly/precisely/
just/-- (N - a number) time/times:
    ....
```

would cause a problem when we tried to call it with

```
follow the throat-clearing rule 2 times.
```

In general, we probably don't need to make our phrase definitions quite so flexible as this, but it's a good idea to account for "a" vs. "the", and for the possibility of using singular and plural forms, especially when writing extensions or other source to be shared.

```
"Ahem"

To do/follow (chosen rule - a rule) exactly/precisely/just
(N - a number) time/times:
```

```
    repeat with index running from 1 to N:
        follow chosen rule.

This is the throat-clearing rule:
    say "'Ahem,' says [a random visible person who is not
the player]."

After waiting:
    do the throat-clearing rule just one time.

Instead of listening:
    follow the throat-clearing rule precisely three times.

Instead of smelling:
    follow the throat-clearing rule exactly 2 times.

Chateau Marmont is a room. Tom, Jack, Zsa-Zsa, and
Wilma-Faye are people in the Chateau. Zsa-Zsa and
Wilma-Faye are women.

Test me with "wait / smell / listen".
```

ExampleFerragamo Again *RB*

Using the same phrase to produce different results with different characters.

Here we use phrases that match individual items where possible, and the general kind otherwise:

"Ferragamo Again"

The Break Room is a room. Vanessa, Tina, and Lisa are women in the Break Room. Mark and Holman are men in the Break Room.

Understand the commands "ask" and "tell" and "answer" as something new.

Understand "talk about [any subject]" as talking about. Talking about is an action applying to one visible thing.

Understand "talk about [text]" as talking randomly about. Talking randomly about is an action applying to one topic. Carry out talking randomly about: say "Mostly you're interested in [the list of subjects]."

Carry out talking about something:
    now the previous subject is the noun.

Report talking about something:
    say "You chat for a while about [the noun]."

A subject is a kind of thing. Assyrian vowel sounds, designer handbags, and instant run-off voting are subjects. Understand "linguistics" and "mute" and "stop" as sounds. Understand "prada" and "tods" and "coach" and "carmen marc valvo" as designer handbags. Understand "reform" and "election" and "election fraud" and "two-party system" and "Diebold" as instant run-off voting.

```
To say (annoyed-person - a person) gestures in
irritation:
    say "[The annoyed-person] sighs heavily. [run
paragraph on]"


To say (annoyed-person - Vanessa) gestures in
irritation:
    say "[The annoyed-person] takes off her glasses and
polishes them on her sleeve. [run paragraph on]".


To say (annoyed-person - Holman) gestures in irritation:
    say "Holman bobs his head. [run paragraph on]"


The previous subject is a subject that varies.


Instead of talking about something for more than one
turn:
    if the noun is the previous subject, say "[a random
visible person who is not the player gestures in
irritation]Maybe you should let this one go.[line break]
[paragraph break]";
    otherwise continue the action.


Test me with "talk about chocolate / talk about vowel
sounds / g / talk about handbags / talk about prada / talk
about tods".
```

ExampleProposal *RB*

Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

Suppose we want to ask the player a question where he might say yes or no in response. There are two possible forms of this: the modal question where the player must pick one to proceed, and the non-modal question

where he might also type other verbs.

"Proposal"

The story genre is "A Worked Example about Yes/No Questions".

Section 1 - Asking a Modal Yes/No Question

When play begins:
    say "Do you like Mr Spruce? ";
    if player consents, now Spruce is handsome;
    otherwise now Spruce is ugly;
    say paragraph break.

Section 2 - Mr Spruce's Non-Modal Question

Use full-length room descriptions.

The Conservatory is a room. "You are in a room full of plants."

Mr Spruce is a man in the Conservatory. Mr Spruce can be apprehensive or calm. Mr Spruce is calm. Mr Spruce can be handsome or ugly.

At 9:02 AM: say "Mr Spruce flings himself to his knees and implores you to become his lawfully wedded wife.";
    now Mr Spruce is apprehensive;

Mr Spruce gives up in two minutes from now.

At the time when Mr Spruce gives up:
say "Mr Spruce sighs heavily, seeing that you don't intend to reply. 'Never mind, my dear, I'll ask later. Perhaps I should have spoken to your Papa first... yes, a gently-bred female... no wonder...'";
now Mr Spruce is calm;
Mr Spruce departs in one minute from now.

At the time when Mr Spruce departs:
if the player can see Mr Spruce, say "Mr Spruce takes his leave of you.";
otherwise say "Mr Spruce pokes his head in to say that he is leaving.";
end the story saying "Well, that is over..."

Instead of saying yes in the presence of an ugly apprehensive Mr Spruce:
now Mr Spruce is calm;
say "Remembering what your mother said to you about the stock exchange and Dear Papa, you close your eyes and accept Mr Spruce.";
end the story saying "Alas for your maiden hopes."

Instead of saying yes in the presence of a handsome apprehensive Mr Spruce:
now Mr Spruce is calm;
say "You are silent with delight for a moment before you say yes, yes!";
end the story saying "How Genevieve Stubbs will cry!"

Instead of saying no in the presence of an ugly apprehensive Mr Spruce:
now Mr Spruce is calm;
say "Gently you inform Mr Spruce that it is impossible. He seems less deflated than you had expected.";

```
    end the story saying "Odd, that..."


Instead of saying no in the presence of a handsome
apprehensive Mr Spruce:
    now Mr Spruce is calm;
    say "You lower your eyes and refuse petulantly,
hoping to stir him to a more ardent repetition of these
same requests. But then -- alack! -- he says 'I see how it
is!' in a strangled voice, and strides from the room!";
    end the story saying "A fatal error!"
```

And since the player might SAY YES TO SPRUCE, we had better reroute the relevant options:

```
Instead of answering Mr Spruce that "no", try saying no.


Instead of answering Mr Spruce that "yes", try saying
yes.


Instead of asking Mr Spruce to try saying yes, try saying
yes.


Instead of asking Mr Spruce to try saying no, try saying
no.


Instead of saying sorry, try saying no.


Instead of asking Mr Spruce to try saying sorry, try
saying no.


Instead of answering Mr Spruce that "sorry", try saying
no.
```

Test me with "z / z / z / yes".

Test more with "z / z / z / no".

---

172 ExampleMatreshka *RB*

A SEARCH [room] action that will open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

"Matreshka"

Ransacking is an action applying to one thing.

Check ransacking:
    if the noun is not the location, say "You can hardly search [the noun] from here." instead.

Carry out ransacking:
    while the player can see a closed openable unlocked container (called target):
        say "[target]: [run paragraph on]";
        try opening the target.

Report ransacking:
    say "You can see nothing further worth searching."

The Russian Gift Shop is a room. In the Russian Gift Shop is a large wooden doll. It is closed and openable.

```
In the large wooden doll is a medium wooden doll. It is
closed and openable. In the medium wooden doll is a
small wooden doll. It is closed and openable. In the
small wooden doll is a tiny solid wooden doll.
```

And now we need to borrow from a later chapter for the command that will make this work:

```
Understand "search [any visited room]" as ransacking.

Test me with "search gift shop".
```

---

173 **Example** **Princess and the Pea** *RB*

The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

The main point here is that we need to figure out where the stack meets the floor:

```
"Princess and the Pea"

The Topmost Turret is a room. A mattress is a kind of
supporter. A mattress is always enterable. A mattress is
portable.

A large mattress is a mattress in the Turret. A medium
mattress is a mattress in the Turret. A small mattress is
a mattress in the Turret.

Instead of sleeping when the player is on a mattress
(called the bed):
```

```
    let the item be the bed;
    while the holder of the item is not a room:
        let the item be the holder of the item;
    say "You can still feel something very uncomfortable
under [the item]."



Instead of sleeping:
    say "You can't sleep standing up!"



Instead of looking under a mattress, say "You scout
around, but are unable to determine what's causing you
this discomfort. If only your maid Winnie were here.
She's very good at this."



Test me with "sleep / enter small / sleep / get up / get
small / put small on medium / get on small / sleep / get
up / g / get medium / put medium on large / get on small
/ look / sleep".
```

ExampleNumberless

RB

A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

```
"Numberless"



The Rambling Warren is a room.



When play begins:
    let N be a random number between 1 and 5;
    if N is 1:
        say "N is one.";
```

```
    otherwise if N is 2:
        say "N is two.";
    otherwise if N is 3:
        say "N is three.";
    otherwise:
        say "N is more than the number of your toes."
```

The final "otherwise" here will fire only if none of the earlier conditions applies; we could leave it out and print nothing in the case that N is 4 or 5.

The more compact way to do this is to create a list of values that our number could match; in many programming languages this is called a switch statement. For example:

```
When play begins:
    let Y be a random number between 6 and 10;
    if Y is:
        -- 6: say "Six is the magic number!";
        -- 7: say "The number of the day is seven!";
        -- otherwise: say "Today's magic number is boring."
```

As a final option, we can use a construction we've seen only briefly before now: a table. The use of tables will be explained more fully in their own chapter, but here we see in brief that we can assign a number of values to one column of a table and then use that table to look up output:

```
When play begins:
    let X be a random number between 11 and 14;
    if X is a number listed in the Table of Switching, say
"[output entry][paragraph break]";
    otherwise say "X is greater than the number of your
noses!"
```

Table of Switching

output	number
"X is eleven!"	11
"X is twelve!"	12
"X is thirteen!"	13

```
Test me with "z".
```

As we shall see, things other than text can be stored in tables, so we could also use a table as a way to look up objects or even rules to carry out.

---

175 Example Wonka's Revenge *RB*

A lottery drum which redistributes the tickets inside whenever the player spins it.

```
"Wonka's Revenge"

The Caribou Lodge is a room. "Hundreds of expectant faces are turned your way from every table." A lottery drum is in the Lodge. "Before you is the lottery drum[if we have spun the drum], ready to disgorge a ticket[otherwise], waiting to be spun[end if]." In the drum are a red ticket, an orange ticket, a yellow ticket, a green ticket, a blue ticket, a purple ticket, and a ticket of pure gold. The drum is closed and openable.

Understand "spin [something]" as spinning.

Spinning is an action applying to one thing.

Check spinning: if the noun is an open container which contains something, say "[The list of things in the noun] would fly out." instead.

Carry out spinning a container:
    shuffle the contents of the noun.
```

```
Report spinning:
    if the noun contains something, say "You rattle [if the
noun is transparent][the list of things in the noun]
[otherwise]the stuff[end if] in [the noun].";
    otherwise say "Nothing results of your shaking [the
noun]."
```

Inform keeps track of the order in which things have been put into a container. If we want to change that order without the player's intervention, we can move the things ourselves.

```
To shuffle the contents of (basket - a container):
    let moves be the number of things in the basket;
    repeat with counter running from 1 to moves:
        move a random thing in the basket to the basket.
```

```
After opening the drum when we have spun the drum for
the first time:
    if something (called the pick) is in the drum:
        try searching the drum;
        say "[The pick] it is, then.";
        silently try taking the pick;
        if the pick is the ticket of pure gold, end the story
finally;
        otherwise end the story saying "Oh well, better luck
next time."
```

```
Test me with "open drum / look in drum / close drum /
spin drum / open drum".
```

ExampleStrictly Ballroom *RB*

People who select partners for dance lessons each turn.

Many simple repetitions can effectively be done with a "now..." instruction: it is quicker to say

```
now every person is angry
```

than

```
repeat with offended party running through people:
    now the offended party is angry.
```

Repeat comes in handy when we have something a bit more complicated to do with each item:

```
"Strictly Ballroom"
```

```
A person can be alert or occupied. A person is usually
alert.
```

```
When play begins:
    now the player is occupied.
```

```
Dance is a kind of value. The dances are waltzes,
polkas, cha-chas, charlestons, fox-trots, tangos,
lambadas, and two-steps.
```

```
The current round is a dance that varies.
```

```
Manner is a kind of value. The manners are swiftly,
primly, energetically, suavely, seductively, elegantly,
and badly.
```

```
Every turn: now the current round is a random dance.
```

```
Every turn:
    repeat with dancer running through people who are
not the player:
        if dancer is alert:
            now dancer is occupied;
            let partner be a random alert person who is not
the dancer;
            if partner is a person:
                now partner is occupied;
                say "[The dancer] [the current round][if a
random chance of 1 in 5 succeeds] [a random manner]
[end if] with [partner]. ";
            otherwise:
                say "[paragraph break][The dancer] is forced
to be a wallflower. Poor [dancer]. ";
    say paragraph break.
```

Notice we did not say "repeat with dancer running through alert people who are not the player...". This is because Inform would draw up a list of alert people at the beginning of the repeat, and not take into account which people became occupied partway through the repetition. If we want to make sure that each person dances only with one other person, we have to continue checking alertness each time we run through the repetition.

After all the partners are assigned, we can set up for the next turn by making everyone alert again, and for this we do not need "repeat":

```
Every turn: now every person is alert; now the player is
occupied.
```

```
Before doing something to someone: now the noun is
occupied.
```

```
Before doing something when the second noun is a
person: now the second noun is occupied.
```

```
Instead of doing something to someone: say "You
successfully distract [the noun]."
```

The Pacific Ballroom is a room. "A rather utilitarian space at the moment, since this is a class and not a party." Timmy, Tommy, Joey, George, Mary, Martha, Yvette, McQueen, Linus, and Patricia are people in the Pacific Ballroom.

Test me with "z / ask linus about blanket / z / z".

ExampleEquipment List *RB*

Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

Most of the phrase options above are relatively self-explanatory; a few are less so. Here is an overview:

"With newlines" tells Inform to put a new line before each listed object. Indented tells it to indent contents of objects, when listing these.

"Giving inventory information" means to append information such as (closed) or (being worn) to objects.

"As a sentence" means to put "and" before the last object and commas between them; this is usually not used in conjunction with newline listing. "As a sentence" obeys whatever conventions about the use of the serial comma we may have established with the "Use serial comma" option.

"Including contents" means to list the contents of open or transparent containers and all supporters, whereas including all contents means to list the contents of all containers, even opaque closed ones.

"Tersely", perhaps unexpectedly, puts parentheses around objects listed as the contents of other objects.

"Giving brief inventory information" omits most of the inventory tags, such as "(open)" and "(worn)", but does list "(closed)" for closed containers which might not otherwise be obviously openable.

"Using the definite article" means prefixing objects with "the", if applicable, rather than "a".

"Listing marked items only" means including only objects that have

already been declared "marked for listing".

"Prefacing with is-are" means that Inform will write "is" before the list if it contains only one item, and "are" if the list contains more than one.

"Not listing concealed items" means to omit from the list anything which is scenery.

Finally, "with extra indentation" means that the whole list should be indented slightly, in emulation of the default inventory listing.

With this information, we can try rewriting the inventory behavior to emulate the standard or to explore alternate versions:

```
"Equipment List"


The Watery Room is a room. The player carries a
snorkel and a waterproof sack. The waterproof sack
contains an undersea map, a diving guide, a cup, and
500 Argentine pesos. The cup contains a worm. The
player wears a swimsuit and a pair of flippers. The sack
is openable and open.


Inventory listing style is a kind of value. The inventory
listing styles are tall, wide, curt, minimal, divided tall,
and divided wide. Current inventory listing style is an
inventory listing style that varies.


Understand "inventory [inventory listing style]" as
requesting styled inventory. Requesting styled inventory
is an action applying to an inventory listing style. It is an
action out of world.


Carry out requesting styled inventory:
    now current inventory listing style is the inventory
listing style understood.
```

```
Report requesting styled inventory:
    say "Inventory listing is now set to [current inventory
listing style]."
```

We begin by emulating the standard inventory listing style:

```
Instead of taking inventory when current inventory listing
style is tall:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying: [line break]";
    list the contents of the player, with newlines,
indented, giving inventory information, including
contents, with extra indentation.
```

Here we offer the alternative of listing everything together as a paragraph:

```
Instead of taking inventory when current inventory listing
style is wide:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying ";
    list the contents of the player, giving inventory
information, as a sentence, including contents;
    say "."
```

This may be unsatisfactory, however. Items that are inside other items are not set off from those merely carried by the player. One way around this is to use terse listing, giving such descriptions as "a waterproof sack (in which are an undersea map, a diving guide, a cup (in which is a worm) and a 500 Argentine pesos)" as opposed to the more confusing " a waterproof sack (open), inside which are an undersea map, a diving guide, a cup, inside which is a worm and a 500 Argentine pesos".

```
Instead of taking inventory when current inventory listing
style is curt:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying ";
    list the contents of the player, tersely, giving brief
inventory information, as a sentence, including contents;
    say "."
```

If, using the above style, we close the sack, we will still get "(closed)" after the sack's listing. The following minimalist listing style abolishes even that nicety:

```
Instead of taking inventory when current inventory listing
style is minimal:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying ";
    list the contents of the player, tersely, as a sentence,
including contents;
    say "."
```

If we want to list worn things separately from carried things, we have occasion to put "listing marked items only" to work:

```
Instead of taking inventory when the current inventory
listing style is divided wide:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are wearing ";
    now all things enclosed by the player are unmarked
for listing;
    now all things worn by the player are marked for
listing;
    if no things worn by the player are marked for listing,
say "nothing";
    otherwise list the contents of the player, as a
sentence, listing marked items only;
    say ".[paragraph break]";
    say "You are carrying ";
    now all things carried by the player are marked for
listing;
    now all things worn by the player are unmarked for
listing;
    if no things carried by the player are marked for
listing, say "nothing";
    otherwise list the contents of the player, as a
sentence, tersely, giving brief inventory information,
listing marked items only;
    say ".[paragraph break]".
```

And similarly for a tall divided inventory:

```
Instead of taking inventory when the current inventory
listing style is divided tall:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    if the player carries something:
        now all things enclosed by the player are
unmarked for listing;
        now all things carried by the player are marked for
listing;
        say "You are carrying: [line break]";
        list the contents of the player, with newlines,
indented, giving inventory information, including
contents, with extra indentation, listing marked items
only;
    if the player wears something:
        now all things enclosed by the player are
unmarked for listing;
        now all things worn by the player are marked for
listing;
        say "You are wearing: [line break]";
        list the contents of the player, with newlines,
indented, including contents, with extra indentation,
listing marked items only.



Test me with "i / inventory wide / i / inventory curt / i /
close sack / i / open sack / inventory minimal / i / close
sack / i / open sack / inventory divided wide / i /
inventory divided tall / i / drop all / i / take all / take off
swimsuit / take off flippers / i / i divided wide / i / wear
swimsuit / drop all / i".
```

Example **M. Melmoth's Duel** *RB*

Three basic ways to inject random or not-so-random variations into text.

```
"M. Melmoth's Duel"



Saint-Germain-des-Prés is a room. "Haunt of artists, of
the coffee-drinking sort, and of cafés, of the artist-
```

haunted sort, you once again find yourself outside M. Melmoth's hotel. Today [one of]the recently-fallen rain runs down the gutters of the 6th[or]sunlight glints even off the blackened windows of the Abbey[or]crowds of vulgar children play chase around the lampposts[at random], and you long to be indoors."

The Hôtel d'Alsace is inside from Saint-Germain-des-Prés. "Typical. Oscar writes you a letter announcing his own imminent demise - 'My wallpaper and I are fighting a duel to the death. One or other of us has got to go.' - and then you get there and he's out, no doubt procuring paint the colour of absinthe, if he isn't procuring the painter."

Tint is a kind of value. The tints are green, aquamarine and darkish purple.

The wallpaper is fixed in place in the Hôtel. The wallpaper has a tint. "In this light, the wallpaper has a distinctly [tint of the wallpaper] wash. [if the tint of the wallpaper is darkish purple]You particularly dislike purple.[end if]"

Before going to the Hôtel: now the wallpaper is a random tint.

After going from the Hôtel, say "You leave, shaking your head. But within twenty-four hours, you are back, as you always knew you would be."

Test me with "in / out / look / in / out / look".

OUT always means "move to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

Suppose we want the game to interpret "GO OUT" as "move towards an outdoors room, or towards a room with more exits than the current room", while "GO IN" means "move toward a room with fewer exits, or towards an indoors room". Thus going in repeatedly within a building would lead towards dead-ends, while going out repeatedly would lead towards the center of the building and then towards an exit to the outside world.

We start by encoding these rules as definitions:

```
"Owen's Law"

A room can be indoors or outdoors. A room is usually
indoors.

Definition: a room is outward:
    if it is not adjacent, no;
    if it is indoors and the location is outdoors, no;
    if it is outdoors and the location is indoors, yes;
    if the number of rooms adjacent to it is greater than
the number of rooms adjacent to the location, yes;
    otherwise no.

Definition: a room is inward:
    if it is not adjacent, no;
    if it is outdoors and the location is indoors, no;
    if it is indoors and the location is outdoors, yes;
    if the number of rooms adjacent to it is less than the
number of rooms adjacent to the location, yes;
    otherwise no.

Instead of going nowhere when the noun is outside: try
exiting.
```

```
Instead of exiting when the player is in a room:
    if at least one room is outward:
        let the destination be a random outward room;
        let the way be the best route from the location to
the destination;
        say "(that is, [way])[command clarification break]";
        try going the way instead;
    otherwise:
        say "It's not entirely obvious which way you mean.
";
        carry out the listing available exits activity.


Instead of going inside when the room inside from the
location is not a room and at least one room is inward:
    if more than one room is inward:
        carry out the listing available exits activity;
    otherwise:
        let the destination be a random inward room;
        let the way be the best route from the location to
the destination;
        say "(that is, [way])[command clarification break]";
        try going the way instead.


Instead of going nowhere:
    carry out the listing available exits activity.
```

This "listing available exits" is a refinement borrowed from a future chapter, which allows us to specify special listing and printing rules:

```
Listing available exits is an activity.


Rule for listing available exits:
    if going inside and an adjacent room is inward:
        say "From here 'in' could reasonably mean [a list of
adjacent inward rooms].";
        rule succeeds;
    if exiting and an adjacent room is outward:
        say "From here 'out' could reasonably mean [a list
of outward adjacent rooms].";
```

rule succeeds;
say "From here you can go [a list of adjacent rooms]."

Before printing the name of a room (called the target) while listing available exits:
let aim be the best route from the location to the target;
say "[aim] to the ".

Rule for printing the name of an unvisited room which is not the location:
say "unknown location".

Dune is an outdoors room. "Hundreds of feet of dune stretch west to the beach, crisscrossed with dune-buggy tracks and the footprints of birds. To the east is a low-lying, boxy concrete installation."

Ocean Shores Military Installation is east of the Dune. It is an outdoors room. "The World War II emplacements, built in case of Japanese invasion, have never been destroyed, though with all the weapons and furnishings gone it is difficult to make much sense of the original structure. A doorway leads west into concrete-lined darkness; a rusty but reliable ladder ascends to a walkway overlooking the sea."

Walkway is above Ocean Shores Military Installation. "From here you have a long view of the dunes and the Pacific Ocean, complete with the rotting hull of a long-stranded vessel."

Dark Echoing Room is inside from Ocean Shores Military Installation. Dank Dripping Room is east of Dark Echoing Room. Narrow Room is south of Dark Echoing Room. Small Sealed Chamber is north of Dark Echoing

Room. Room Smelling of Animal Urine is north of Dank Dripping Room. The description of a room is usually "It is dark in here, and feels unsafe."

Test me with "e / u / d / in / s / out / n / out / e / in / out / out / out".

---

180 **Example** **Witnessed 2** *RB*

A piece of ghost-hunting equipment that responds depending on whether or not the meter is on and a ghost is visible or touchable from the current location.

"Witnessed"

The player carries a device called a Trifield Natural EMF Meter. The description of the Meter is "This cost a pretty penny off the internet, but it's worth it: according to the website it has been programmed by PhD physicists to ignore manmade sources of fields and to respond only to paranormal EMF changes.

It also features an optional Tone Alarm, which can be turned on to indicate when readings spike. If the alarm is off, the meter just reads out the magnetic and electric field levels on a scale from 0-100 microteslas, or 0-1000 V/m.

Since both fields are important, you keep the meter set to SUM mode. The meter has its own optional backlighting, so that you can see the reading even if your flashlight is off. Currently it is reading at [meter setting]." A Tone Alarm is part of the Meter. It is a device. The description of the Tone Alarm is "The Tone Alarm will make a noise, if the EMF picks up a spike."

To decide what number is meter setting:
    if the meter is switched off, decide on 0;
    if a ghost is touchable, decide on 35;
    if a ghost is visible, decide on 12;
    decide on 0.

After switching on the meter:
    say "You turn on the meter. The needle steadies at [meter setting]."

Every turn: if the meter setting is greater than 10 and the Tone Alarm is switched on, say "[The Tone Alarm] shrieks."

Thirtieth Street Station is a room. "A huge, high, rectangular room with coffered ceilings, which looks grand but mostly makes you feel lonely and small. There are long benches in rows down the middle of the room, and an information desk with the train times, and a series of ticket windows, none of which matters very much at the moment."

The benches are an enterable supporter. They are scenery in the Station. The information desk is scenery in the Station. Some ticket windows are scenery in the Station. Instead of examining scenery in the Station: say "You're fairly sure that whatever is going on here has nothing to do with [the noun]." Understand "window" as ticket windows.

The mural is fixed in place in Thirtieth Street. "At the north side of the station is a particularly pointless and empty annex to the main room. It is dominated by a huge relief of sorts, and this is what you remember." Understand "metal" or "relief" or "huge" as the mural. The description of the mural is "It is both stylized and confusing, but you think it might be supposed to

represent the various tasks and occupations of Philadelphia's population. The portions closer to the ground look as though they have recently been subjected to a light dusting of talcum powder. No unusual prints are evident."

The wind chimes are fixed in place in Thirtieth Street. "Carefully attached to the wall with a piece of duct tape and a hook is a light-weight set of wind chimes. Someone else has been here before you, it seems." The description is "Several of your friends use wind chimes as a sort of ghost alarm, since ghosts sometimes cause very localized movements of air when there is no natural breeze."

A ghost is a kind of person. The pale figure is a ghost.

At 9:03 AM: move the pale figure to the location; say "You shiver with some sort of presence."

Test me with "turn on alarm / turn on meter / z / z / z / x figure".

ExampleA Haughty Spirit *RB*

Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

Suppose we have a game in which the player can climb through windows which overlook rooms below. We want him to be allowed to climb out windows to reach a room on the same level or at most one level lower than the one he's on; otherwise, he should get a refusal, saying that he would break his neck.

To figure out the height distance between the start room and the destination room, we might have a repeat loop look at all the directions one has to follow along the "best route" path between the two rooms, and

record any ups and downs; then subtract the number of "up" steps from the number of "down" steps, and report what remains.

```
"A Haughty Spirit"


To decide what number is the distance (first place - a
room) rises above (second place - a room):
    let the total distance be the number of moves from
the first place to the second place;
    if the total distance is less than 1, decide on 0;
    let count of down moves be 0;
    let count of up moves be 0;
    let next place be the first place;
    repeat with counter running from 1 to the total
distance:
        let the way be the best route from the next place to
the second place;
        if the way is down, let count of down moves be the
count of down moves plus 1;
        if the way is up, let the count of up moves be the
count of up moves plus 1;
        let next place be the room the way from next place;
    let the decision be the count of down moves minus
the count of up moves;
    decide on the decision.
```

Now we just have to create windows and some action rules for interacting with them...

```
A window is a kind of thing. A window is always fixed in
place. A window can be open or closed. A window is
usually closed. A window can be openable or
unopenable. A window is usually openable.


Understand "climb through [something]" as entering.
Understand "jump through/out [something]" as entering.


Before entering a closed window:
    say "[The noun] would have to be opened first."
```

```
instead.


Instead of entering a window:
    if the noun overlooks a room (called the far side):
        let fall be the distance the location rises above the
far side;
        if fall is greater than 1, say "You'd break your
neck." instead;
        say "You tumble into [the far side].";
        move the player to the far side;
    otherwise:
        say "There's nowhere to go."


Instead of examining a window:
    say "[The noun] [if the noun is open]opens
over[otherwise]gives a view of[end if] [the list of rooms
overlooked by the noun]."
```

Here we must anticipate a little from the chapter on Relations, and provide ourselves with a way of keeping track of how windows and rooms relate to one another:

```
Overlooking relates various windows to various rooms.
The verb to overlook means the overlooking relation.
The initial appearance of a window is usually "[The item
described] overlooks [the list of rooms overlooked by the
item described]."


The Square Keep is above the Winding Staircase. The
Winding Staircase is above the Motte. A crown and a
broken sword are in the Motte. The Bailey is west of the
Motte.


The long window is in the Keep. The long window
overlooks the Bailey and the Motte. The narrow window
is in the Winding Staircase. The narrow window
overlooks the Bailey.
```

```
Test me with "jump through window / open window /
jump through window / d / x narrow window / open
window / climb through window / e / up / down".
```

We could then add rules to allow the player to look through windows and see things in the rooms below, but that would require more material from later chapters.

---

182 **ExampleEntropy** *RB*

All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

```
"Entropy"

Heat is a kind of value. The heats are frosty, cold, cool,
room temperature, warm, hot, and scalding. Everything
has a heat. The heat of a thing is usually room
temperature.

Every turn:
    repeat with item running through things which are not
in an insulated container:
        if the heat of the item is greater than room
temperature, now the heat of the item is the heat before
the heat of the item;
        if the heat of the item is less than room
temperature, now the heat of the item is the heat after
the heat of the item.

Definition: a container is insulated if it is closed and it is
opaque.
```

The vacuum thermos is an opaque closed openable container carried by the player. In the vacuum thermos is a frosty thing called an ice cube.

Every turn:
    if the heat of the ice cube is greater than cold:
        if the ice cube is visible, say "The ice cube melts! 'HA ha,' says Maxwell, in a very unsporting, some might say demonic, way.";
        now the ice cube is nowhere.

Before printing the name of something: say "[heat] ".

Equilibrium is a room. "A perfectly smooth chamber sealed from the outside world. You can't at this moment work out where the exit is, though possibly that is just because the lighting is so very very even and diffuse. And doesn't come from anywhere that you can see, either."

Maxwell is a man in Equilibrium. "Maxwell perches awkwardly on a stool across from you[if Maxwell has something], holding [a list of things carried by Maxwell][end if]." He is carrying a box of Chinese food. The Chinese food is scalding. "A discarded [item described] lies on the floor." The description of Maxwell is "He has the faintly peevish look of one who has not been properly fed."

Every turn when Maxwell has the food:
    if the heat of the Chinese food is greater than warm, say "Maxwell takes a bite, and swears.";
    if the heat of the Chinese food is warm, say "Maxwell eats as fast as he can, enjoying the food while it's at just the right temperature.";
    if the heat of the Chinese food is less than warm:
        say "Maxwell sadly stabs at his leftovers with a chopstick, but does not try to eat any more.";
        move the food to the location.

Test me with "z / z / open thermos / close thermos / open thermos".

Example **The Hang of Thursdays** *RB*

Turns take a quarter day each, and the game rotates through the days of the week.

"The Hang of Thursdays"

The Stage is a room. Rule for printing the name of the stage: say "[current weekday] [current time period]" instead.

A weekday is a kind of value. The weekdays are Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday. The current weekday is a weekday that varies. The current weekday is Saturday.

A time period is a kind of value. The time periods are morning, afternoon, evening, night. The current time period is a time period that varies. The current time period is afternoon.

This is the new advance time rule:
    if the current time period is less than night:
        now the current time period is the time period after the current time period;
    otherwise:
        now the current time period is morning;
        now the current weekday is the weekday after the current weekday.

Now we need to borrow from a later chapter to make these instructions apply to the passage of time:

```
The new advance time rule is listed instead of the
advance time rule in the turn sequence rules.


Test me with "z / z / z / z / z".
```

Chapter 12: Advanced Actions

aking actions work for other people; *§12.12. Check rules for actions by other people*; *§12.13. Report rule*

Contents of *Writing with Inform*
Chapter 11: Phrases
Chapter 13: Relations
Indexes of the examples

§12.1. A recap of actions

Actions are impulses to do something, which arise sometimes through typed commands:

```
>examine tapestry
```

and sometimes through "try" phrases occurring in other rules:

```
Before examining the tapestry, try switching the ultraviolet light on.
```

Every action either succeeds or fails, though failure may not be a bad thing (something better may have happened). Besides any rules applied in the source text, actions are subject to basic realism rules. A general rule ensures that actions are rejected if the actor would need to touch something which is out of reach, or see something which is invisible; and a couple of hundred other built-in rules police individual actions. For instance, if the ACTIONS testing command has been used to switch monitoring on, then:

```
>unlock cage with watermelon
[unlocking cage with watermelon]
That doesn't seem to fit the lock.
[unlocking cage with watermelon - failed the can't unlock without the
correct key rule]
```

Actions generated by "try" phrases are allowed to run "silently", which means that if nothing out of the way happens and they succeed, then nothing is printed. For

instance:

```
Before examining the tapestry: say "(Switching on the lamp first.)";
silently try switching the ultraviolet light on.
```

There are many ways to impose extra rules on actions, and we have seen three main kinds: Before rules, intended so that preliminary activities like the one above can happen before the action is tried; Instead rules, which block or divert the intention, or may cause something spectacularly different to happen; and After rules, which allow for unexpected consequences after the action has taken place.

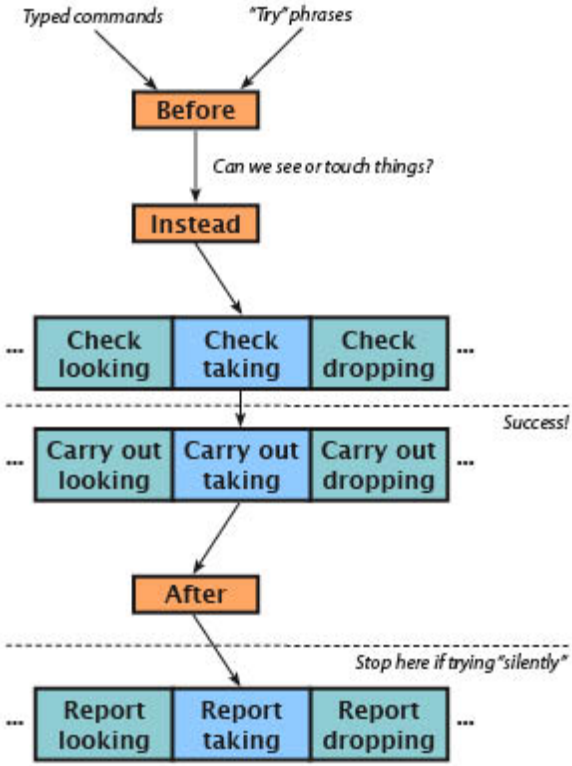
Start of Chapter 12: Advanced Actions

Back to Chapter 11: Phrases: §11.18. The value after and the value before

Onward to §12.2. How actions are processed

§12.2. How actions are processed

The following flow chart shows the natural course of events when Inform deals with a new action - a "taking" action in the case drawn. For quite a long time, the action may still fail, and it may be that nothing actually happens: but eventually a deciding line is crossed, and once that happens the action will certainly succeed.



The coloured boxes on this chart represent "rulebooks", that is, collections of rules with a common purpose. The orange boxes for Before, Instead and After were covered in the Basic Actions chapter, but the blue boxes are new. The orange boxes are where we put rules to handle unusual situations, or unexpected events: special rules to cover the opening of a container which happens to be booby-trapped, or walking through a doorway into a room where a surprise party is about to be sprung.

Blue boxes hold the mundane, everyday rules - the generic ways that particular actions behave. Every action provides these: "Check" rules, to see if it makes sense - for instance, to see that the player is not trying to take his or her own body, or a whole room, or something he or she already has; then "Carry out" rules, to actually do what the action is supposed to do - in the case of taking, to move an object into the player's possession; and finally "Report" rules, which tell the player what has happened - perhaps printing up "Taken."

When we create a new action, we add a new column to the blue rows in this diagram. As we shall see, we can also put new rules into the existing blue boxes: for instance, if we wanted to increase physical realism by forbidding the player to carry more than a certain weight, we would want to add a new "check taking" rule, and this is entirely legal.

In subsequent chapters, we will see ways to intervene at almost every point in the above diagram - from how "Can we see or touch things?" is reckoned, to each and every individual rule in all of these books. Action-processing may be the single most important thing Inform does, so the system is designed to be immensely flexible. On the other hand, that does make it a lot to take in at one look. Newcomers should probably concentrate on "Instead" and "After" as the basic tools for designing the situations turning up routinely in interactive fiction. There are guidelines at the end of this chapter offering advice on which tricks to use when it comes to more complicated needs.

Start of Chapter 12: Advanced Actions
Back to §12.1. A recap of actions
Onward to §12.3. Giving instructions to other people

§12.3. Giving instructions to other people

So far, all actions have been carried out by the player: which is fine for exploring the passive world of an empty warehouse, but less good for a drama in which other characters have to be contended with. In fact, an action can be carried out by anybody - by any instance of the "person" kind, that is, which includes all the men, women and animals in the story, and not only the player.

In interactive fiction, players conventionally ask other characters to do something with commands like so:

```
> will, go west
```

Clearly "will, go west" should not produce the same action as "go west", because a different person will be trying it: this person is called the "actor", and while the actor is ordinarily the player, here it is the character called Will. Inform distinguishes these two actions like so:

```
going west
asking Will to try going west
```

As a result, we can write rules like so:

```
Instead of asking Will to try going west, say "Will scratches his head,
baffled by this talk of westward. Is not the ocean without bound?"
```

To write rules like this, we sometimes want to generalise about who is supposed to do the deed. To do this we can refer to "person asked", just as the "noun" stands for whatever noun was typed:

```
Instead of asking somebody to try taking something, say "I don't think
we ought to tempt [the person asked] into theft, surely?"
```

So if the player types "Algy, take sandwich", the "person asked" would be Algy; the "noun" would be the sandwich; and there would be no "second noun".

Start of Chapter 12: Advanced Actions
Back to §12.2. How actions are processed
Onward to §12.4. Persuasion

Example 184: **Virtue** Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

Example 185: **Latris Theon** A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

---

§12.4. Persuasion

"Asking ... to try ..." actions run through their Before and Instead rules like any other actions, but then (if no rule has intervened) something different happens: Inform has to decide whether the person asked consents to try the action or not. By default, the answer is always no, and text like the following will be printed:

```
> will, go west
Will has better things to do.
```

However, we can intervene to make the answer "yes", using a special kind of rule which produces a yes/no answer. The following examples show how we can give broad or narrow permission, as we choose:

```
Persuasion rule for asking people to try going: persuasion succeeds.
Persuasion rule for asking Will to try going west: persuasion
succeeds.
```

Such a rule can either declare that "persuasion succeeds", or that "persuasion fails", or make no decision and leave it to another rule to say. If it decides that persuasion fails, it is also allowed to say something, describing why: in that event, the standard message ("Will has better things to do.") is suppressed. For example,

```
Persuasion rule for asking Will to try going:
    say "Will looks put out, and mutters under his breath.";
    persuasion fails.
```

The following rule, which is really only suitable for testing, makes everybody infinitely obliging:

```
Persuasion rule for asking people to try doing something: persuasion
succeeds.
```

Supposing that Will does decide to cooperate, a new action is generated:

```
Will going west
```

and this is then subject to all of the usual action machinery. For instance, we could write a rule such as:

```
Instead of Will going west, say "He runs out into the waves, but soon
returns, rueful."
```

So in this case the new action ("Will going west") failed: but the original action, "asking Will to try going west", is still deemed to have succeeded - after all, Will *did* try. To put it more formally, "asking X to try A" succeeds if the persuasion rules

succeed, and otherwise fails.

Note also that "Instead of..." rules written for other people will be treated by Inform as failures, even if we write something like

```
Instead of Will pulling the cord:
    say "The bell rings."
```

and thus may produce unsatisfactory results such as

```
>WILL, PULL CORD
The bell rings.

Will is unable to do that.
```

If we wish to write new successful actions for another character, we will need to create appropriate carry out and report rules for them: these will be explained in the sections to follow.

(Finally, note that the mechanism Inform uses to see if we have printed a refusal message of our own, in the event of persuasion rules failing, can be fooled if we write a persuasion rule explicitly ending with a "[paragraph break]" text substitution.)

Start of Chapter 12: Advanced Actions

Back to §12.3. Giving instructions to other people

Onward to §12.5. Unsuccessful attempts

Example 186: ★ **The Hypnotist of Blois** A hypnotist who can make people obedient and then set them free again.

Example 187: ★ **Police State** Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

§12.5. Unsuccessful attempts

Suppose, finally, that Will not only consents to try the action, but it also survives its passage through Before and Instead rules. What happens then? In principle, what happens to Will is exactly what would have happened to the player in his place. For instance:

```
> will, go east
Will leaves to the east.
```

If on the other hand Will's attempt is frustrated because one of the checking rules stops him, then Will's action fails. For instance, if Will tries going northeast but there is no room to northeast, one of the rules checking the "going" action will stop him. We will then see this:

```
> will, go northeast
Will is unable to do that.
```

This is rather a generic message, and we may want something more interesting. We can provide that using yet another special kind of rule:

```
Unsuccessful attempt by Will going: say "Will blunders around going
nowhere, as usual."
```

Even that is still a little generic, though, because it treats all of the various ways that "going" can fail as the same. If we have ACTIONS switched on, we can see what goes on behind the scenes when we ask Will to walk into a door:

```
>will, go west
[asking Will to try going west]
[(1) Will going west]
[(1) Will going west - failed the can't go through closed doors rule]
Will blunders around going nowhere, as usual.
[asking Will to try going west - succeeded]
```

(The "(1)" lets us know that a new action is starting during the old one, and before the old one finishes: sometimes we go up to three or four deep, though seldom more in practical cases.) We can now rewrite the "unsuccessful attempt" rule like so:

```
Unsuccessful attempt by Will going:
    if the reason the action failed is the can't go through closed doors
rule, say "Will looks doubtful and mumbles about doors.";
    otherwise say "Will blunders around going nowhere, as usual."
```

The value "reason the action failed" is set to whichever checking rule threw out the action which Will tried. The names of these rules try to be self-explanatory - at any rate, those with gnomic names are not useful for this sort of thing, and can be ignored - and can be found out either using ACTIONS or by consulting the Actions index.

Finally, note that "unsuccessful attempt" rules apply only when the person in question is being asked to perform the action by somebody else - as in the examples above.

Start of Chapter 12: Advanced Actions
Back to §12.4. Persuasion
Onward to §12.6. Spontaneous actions by other people

Example 188: ★★ **Generation X** A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

---

§12.6. Spontaneous actions by other people

The player's actions happen not only when he types a command, but can also happen spontaneously as a result of a "try" phrase.

```
try going west
try asking Will to try going west
```

The latter might, of course, result in Will trying going west: or it might not - that depends on the persuasion rules. But as the author, we have the ultimate powers of persuasion, and can make Will act in any way we like, without asking:

```
try Will going west
```

Nobody in the simulated world requested this: it is an impulse felt by Will alone, so that - from the player's point of view - Will is acting spontaneously. The player need not be anywhere nearby, and may never know what happened. Recall that when actions work their way down through the flow-chart, they are stopped before reaching the "report" stage - when the player is told about them - if they are running "silently". This is also where Inform stops an action which is not witnessed by the player.

To repeat a point in the previous section: "unsuccessful attempt" rules do not apply to actions which the author has caused to happen, using "try". When such actions fail, they invoke no special set of rules. Indeed, when "try" causes somebody other than the player to try an action, nothing will be printed to report back on success or failure. If Will can't go west, that's his problem.

Note that the text "try Will going west" involves the actor's name immediately placed next to the action he is to try, which in a very few cases might cause ambiguities. If the actor's name contains a participle like "going" - say, if Will's full name turned out to be Mr Will Going - then we would have to write out the action name in full, using "trying" to clarify matters:

```
try Will Going trying going west
```

Start of Chapter 12: Advanced Actions
Back to §12.5. Unsuccessful attempts
Onward to §12.7. New actions

Example 189: **IQ Test** Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

Example 190: **Boston Cream** A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

---

§12.7. New actions

It is not often that we need to create new actions, but a large work of interactive fiction with no novelty actions is a flavourless dish. Here we shall create an action for photographing things.

```
The Ruins is a room. "You find this clearing in the rainforest oddly
familiar." The camera is in the Ruins. "Your elephantine camera
hangs from a convenient branch."


Photographing is an action applying to one visible thing and requiring
light.
```

In theory that text is already sufficient to make the new action, but what we have so far is rudimentary to say the least. The two qualifications give Inform the useful information that we cannot photograph in the dark, and that we need to be photographing something - not, as in the case of waiting or taking inventory, acting without reference to any particular thing external to ourselves.

The word "visible" here tells Inform that we do not need to be able to touch the thing in question: a line of sight is good enough. These two stipulations were necessary because the default arrangement is that any object must be in touching range, and that most actions can be performed in darkness. (Also, note that if you invent an action which needs to apply to directions like "north" or "south", you need to make this apply to visible things, because the object used inside Inform to represent the idea of "north" can be seen but not touched. So for understanding purposes, "visible thing" is understood as meaning any visible thing or direction: it's more general than "thing", not more specific.)

Occasionally, when writing general rules about actions, it can be useful to find out what the current action's requirements are: the following conditions do what they suggest.

if action requires a touchable noun:

This condition is true if the action being processed is one whose (first) noun is an object which needs to be touchable by the actor. For example, it's true for "taking", but false for "examining".

if action requires a touchable second noun:

This condition is true if the action being processed is one whose second noun is an object which needs to be touchable by the actor. For example, it's true for "putting the brick in the sack", but false for "throwing the brick at the window".

if action requires a carried noun:

This condition is true if the action being processed is one whose (first) noun is an object which needs to be carried by the actor. For example, it's true for "dropping", but false for "taking".

if action requires a carried second noun:

This condition is true if the action being processed is one whose second noun is an object which needs to be carried by the actor.

if action requires light:

This condition is true if the action being processed is one which can only be performed if the actor has light to see by. For example, it's true for "examining", but false for "dropping".

As further examples, here we create "blinking" and "scraping X with Y". Note the use of "it" to indicate that the name of an object should go here.

```
Blinking is an action applying to nothing. Scraping it with is an action applying to two things.
```

The photographing action now exists, but with two provisos: (a) it never happens, because Inform does not know what commands by the player should cause it, and (b) even if it were to happen, nothing would follow, because Inform does not know what to do. (There are no check, carry out or report rules yet.)

The first problem is easily overcome:

```
Understand "photograph [something]" as photographing.
```

We will return to the whole subject of parsing, as this process of understanding the player's commands is called, later. But this gives the gist of it.

See **Understand** *for the full story*

Start of Chapter 12: Advanced Actions

Back to §12.6. Spontaneous actions by other people

Onward to §12.8. Irregular English verbs

Example 191: **Red Cross** A DIAGNOSE command which allows the player to check on the health of someone.

Example 192: **Frizz** Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

Example 193: **3 AM** A shake command which agitates soda and makes items thump around in boxes.

---

§12.8. Irregular English verbs

Our three example actions can be recognised in play using the following:

```
Understand "photograph [something]" as photographing.

Understand "blink" as blinking.

Understand "scrape [something] with [something]" as scraping it with.
```

The last of these examples shows why Inform does not risk generating this automatically: English is so full of irregular verbs. Inform could have guessed "blink" and "photograph", but might then have opted for "scrap" instead of "scrape".

Inform does risk automatically generating the past participle of an action. (Many past participles are never needed, so the stakes are lower if Inform gets this wrong.) What usually happens is that the "-ing" is replaced with "-ed", thus photographed, blinked, scraped - but Inform has a dictionary of some 460 irregular exceptions, such as caught, fled, crossbred, taken, woven. So with luck Inform will guess correctly. If not, we can get around this like so:

```
Squicking is an action with past participle squacked, applying to one thing.
```

Start of Chapter 12: Advanced Actions
Back to §12.7. New actions
Onward to §12.9. Check, carry out, report

§12.9. Check, carry out, report

The normal behaviour of an action is specified by its three associated rulebooks - check, carry out and report. In the case of our "photographing" example, these rulebooks will be:

Check photographing. Here, for instance, we need to verify that the player has the camera. If any of our checks fail, we should say why and stop the action. If they succeed, we say nothing.

Carry out photographing. At this stage no further checking is needed (or allowed): the action now definitively takes place. At this point we might, for instance, deduct one from the number of exposures left on the film in the camera, or award the player points for capturing something interesting for posterity. But we should say nothing.

Report photographing. At this stage no further activity is needed (or allowed): whatever effect the action had, it has happened and is now over. All we can do is to say what has taken place.

So far we have not really gone into the business of what rulebooks are, and we don't do so here either - suffice to say that we can now create whatever rules we need:

```
A check photographing rule:
    if the camera is not carried:
        say "You can hardly photograph without a camera, now can
you?" instead.
```

In fact, writing "a check photographing rule" is over-formal. We can more simply label our rules like so:

```
Check photographing:
    if we have photographed the noun:
        say "You've already snapped [the noun]." instead.


Report photographing: say "Click!"
```

For the sake of brevity, photography has no interesting consequence (no points to be won, no film to use up), so there are no carry out rules here. Note the way we used the word "instead" once again to stop actions in their tracks.

We can continue to add rules at any point, and a classic thing that happens when testing a new work is that the designer realises there is a case which has not been thought of:

```
Check photographing:
    if the noun is the camera:
        say "That would require some sort of contraption with mirrors."
instead.
```

Start of Chapter 12: Advanced Actions
Back to §12.8. Irregular English verbs
Onward to §12.10. Action

variables

Example 194: ★ **The Dark Ages Revisited** An electric light kind of device which becomes lit when switched on and dark when switched off.

Example 195: ★★ **Paddington** A CUT [something] WITH [something] command which acts differently on different types of objects.

Example 196: ★★★ **Delicious, Delicious Rocks** Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

Example 197: ★★★ **Noisemaking** Creating a stage after the report stage of an action, during which other characters may observe and react.

---

§12.10. Action variables

For some complex situations, it can be useful to keep track of a few values throughout the processing of the action. This is not an everyday occurrence: in the Standard Rules, for instance, only two or three out of 90 actions need to do this. But suppose we want to write a more deluxe version of our "photographing" action. This time, rather than having a single thing called the "camera", we will provide a whole range of possible cameras, varying in quality:

```
Photographing is an action applying to one visible thing and requiring
light. Understand "photograph [something]" as photographing.
```

```
The Studio is a room. Sally is a woman in the Studio. A foam-lined
tote bag is in the Studio.
```

```
A camera is a kind of thing. A camera has a number called picture
quality. The digital SLR camera is a camera in the tote bag. The
player carries a camera called the instant one-shot camera. The
picture quality of the SLR camera is 10. The picture quality of the
one-shot is 2. Definition: a camera is sharp if its picture quality is 5 or
more.
```

And we will want the photographing action to have the player use the best-quality camera which comes to hand. We will give the action a variable called the 'camera photographed with', thus:

```
The photographing action has an object called the camera
photographed with.
```

Every action's variables must be named differently from those of all other actions, because there are some "before" rules (for instance) which take effect for many different actions, and which might need access to any of their variables. So action variables should be named in a way marking out to which action they belong. The best way to do this is to include the past participle of the action name - just as "camera photographed with" contains the past participle "photographed" of the action "photographing".

This value is created when the action begins, and disappears when the action ends. (If the action should happen a second time before the first time was completed, a second copy of the value is created, leaving the original undisturbed.) When the action begins, the value starts out as something neutral - so if it is a number, it starts out as 0, if a text, it starts out as the blank text "", and so on. Here it is an object, so it starts out as nothing - the value meaning no object at all. But of course we want to give it a value ourselves. We can do that using the "setting action variables" rulebook. For instance:

```
Setting action variables for photographing:
    now the camera photographed with is the sharpest camera which
is carried by the actor.
```

The "setting action variables" rulebook is run through before even the before rules, and it has no power to stop or change the action. Its rules should say nothing and do nothing other than to set rulebook variables like this one. Note that it is intended to work for any actor, not only the player: so rather than referring to the player as the performer of the action, we need to write "the actor", as in the example above. (See

subsequent sections for more on actors.)

We can now write rules such as:

```
A check photographing rule:
    if the camera photographed with is nothing:
        say "You can hardly photograph without a camera, now can
you?" instead.
```

Only rules to do with the photographing action - before, instead, after, check, carry out, or report rules, and so on - are allowed to see the 'camera photographed with' value: it's the private property of the action.

A further elaboration allows us to make rules about photographing neater to write. If we create our variable like so:

```
The photographing action has an object called the camera
photographed with (matched as "using").
```

...then we are now allowed to add an optional 'using ...' clause onto a description of the action. The clause has to be introduced with a single word: here, it's 'using'. For instance, we could write rules such as

```
Instead of photographing something using the one-shot camera:
    say "But you promised to give this to Sally's nephew."

Check photographing something using the noun:
    say "That would require some sort of contraption with mirrors."
instead.

Report photographing something using a sharp camera:
    say "You feel cool and important as the shutter clicks."
```

(This is the method used by the Standard Rules to attach optional clauses such as 'to', 'with' and 'through' to the going action.)

Back to §12.9. Check, carry out, report
Onward to §12.11. Making actions work for other people

Example 198: ★ **Removal** TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

Example 199: ★ **Further Reasons Why All Poets Are Liars** The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

Example 200: ★ **The Second Oldest Problem** Adapting the going action so that something special can happen when going from a dark room to another dark room.

Example 201: ★★ **Puff of Orange Smoke** A system in which every character has a body, which is left behind when the person dies; attempts to do something to the body are redirected to the person while the person is alive.

Example 202: ★★★ **Croft** Adding special reporting and handling for objects dropped when the player is on a supporter, and special entering rules for moving from one supporter to another.

§12.11. Making actions work for other people

The "photographing" action now works very nicely when the player does it. But not when others try. Suppose that neither the player, nor Clark Gable, is holding the camera:

```
>photograph clark
You can hardly photograph without a camera, now can you?

>clark, photograph me
>
```

An uncanny silence. What has happened is that the rules written so far are all implicitly restricted to the player only. This is because when we write -

```
Check photographing:
    if the camera is not carried:
        say "You can hardly photograph without a camera, now can
you?" instead.
```

the action is "photographing", not "Clark photographing". In the next few sections we shall see how to make the rules work nicely for everybody. This is a little bit harder, so it should be noted right away that in many projects there is no need. In a story which has no other characters who succumb to persuasion, for instance, only the player will ever try the action.

Start of Chapter 12: Advanced Actions

Back to §12.10. Action variables

Onward to §12.12. Check rules for actions by other people

Example 203: ★★ **The Man of Steel** An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

Example 204: ★★★ **Trying Taking Manhattan** Replacing

the inventory reporting rule with another which does something slightly different.

Example 205: ★★★★ **Under Contract** Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

---

§12.12. Check rules for actions by other people

If we want to impose the restriction about carrying the camera on other people, we need a rule like the following:

```
Check someone photographing: if the person asked does not carry
the camera, stop the action.
```

Implicitly, that "someone" excludes the player. Note that we say nothing in this rule, stopping the action without a word: after all, Clark might well be out of sight when trying this. If he is within sight, then we read:

```
>clark, photograph me
Clark Gable is unable to do that.
```

We saw before that Inform's built-in rules all have handy names (the "can't drop what's already dropped rule", and such), and that these are useful when writing better "unable to..." messages. So for a deluxe version, we end up with:

```
Check someone trying photographing (this is the other people can't
photograph without the camera rule): if the person asked does not
carry the camera, stop the action.
```

And now, with ACTIONS on, we find that:

```
>clark, photograph me
```

```
[asking Clark Gable to try photographing yourself]
[(1) Clark Gable photographing yourself]
[(1) Clark Gable photographing yourself - failed the other people can't
photograph without the camera rule]
Clark Gable is unable to do that.
[asking Clark Gable to try photographing yourself - succeeded]
```

which means that we could have, say,

```
Unsuccessful attempt by Clark photographing:
    if the reason the action failed is the other people can't photograph
without the camera rule, say "Clark is too suave to be embarrassed.
'Frankly, my dear, I don't have a camera.'";
    otherwise say "Clark tries, and fails, to take a photograph."
```

Start of Chapter 12: Advanced Actions

Back to §12.11. Making actions work for other people

Onward to §12.13. Report rules for actions by other people

Example 206: **Get Axe** Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

Example 207: **Barter** **Barter** Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

---

§12.13. Report rules for actions by other people

Report rules for the player's actions are easy to write, and for many actions, they are not much harder for other people either:

```
Report photographing: say "Click!"
Report someone photographing: say "Click! [The person asked] takes
a snapshot of [the noun]."
```

But once other people are involved, we have to go to some trouble to get all of the possibilities right. Here is a case which did not immediately occur to the author of the "going" action, for instance:

```
>get in cage
You get into the cage.
>clark, get in automobile
Clark Gable gets into the automobile.
>clark, push cage west
Clark Gable goes west in the automobile, pushing the cage in front,
and you along too.
```

```
The Lot (in the cage)
In the Lot you can see an automobile (in which is Clark Gable).
```

We said before that report rules are skipped if the action is running "silently", or if the action is one that the player does not witness. But that is also a tricky concept. Inform's doctrine is that you witness an action if you can see any of the actor, the noun or the second noun at either the beginning or the end of the action; except that being able to see a backdrop does not count. Thus if Clark Gable, in Beverly Hills, photographs the Hollywood sign then we do not witness this from Sunset Boulevard merely because we, too, can see the Hollywood sign.

While the report rules for actions by the player must actually report something, report rules for other people's actions are under no such obligation. For instance, if Clark unlocks a door from the other side to the player, then this counts as an action that the player witnesses - and after all, it could be argued that the player should hear the key turning in the lock - but in fact the standard rules for reporting locking choose to say nothing.

Start of Chapter 12: Advanced Actions

Back to §12.12. Check rules for actions by other people

Onward to §12.14. Actions for any actor

Example 208: ★★ **The Man of Steel Excuses Himself**
Elaborating the report rules to be more interesting than "Clark goes west."

Example 209: ★★★ **Fate Steps In** Fate entity which attempts to make things happen, by hook or by crook, including taking preliminary actions to set the player up a bit.

---

§12.14. Actions for any actor

In the previous sections, we created a new action by providing one set of rules for the player and another for anybody else who might try to perform it. These rules began with action descriptions in one of the following forms:

```
Instead of taking a container, ...
Instead of P taking a container, ...
```

The first form implies that the player must be performing the action: the second allows for any person matching P to be the action, except that this person must not be the player. That means that all rules seen so far either affect only the player, or only other people.

This is often convenient, but sometimes we need to set up a complicated action which really does work in the same way for every actor - for instance, the built-in Inform actions provided by the Standard Rules aim to do this. We can write such rules thus:

```
Instead of an actor taking a container, ...
```

Here the rule applies to anyone who tries taking a container, player or not. Inside such a rule, the special value 'the actor' is the person performing the action. For instance, the Standard Rules include this one:

```
Carry out an actor wearing (this is the standard wearing rule):
```

```
now the actor wears the noun.
```

Start of Chapter 12: Advanced Actions
Back to §12.13. Report rules for actions by other people
Onward to §12.15. Out of world actions

§12.15. Out of world actions

The actions seen so far are all impulses causing the protagonist inside the fictional world to do something, or at least try to. But when the player types "quit" or "save", that is not a request for anything to happen in the fictional world: it is an instruction to the program simulating that world. In fact, just the same, such requests are treated as actions, but of a special category called "out of world" actions. They do not cause time to pass by, so the turn counter does not advance, nor does this command cycle count as a turn at all; and they are altogether exempt from "Before", "Instead" and "After" rules. Only the player is allowed to try them.

We can also create new out-of-world actions. Suppose we want a dialogue like so:

```
>ROOMS
You have been to 1 out of 8 rooms.
```

Here is a complete implementation:

```
Requesting the room tally is an action out of world.
Report requesting the room tally: say "You have been to [number of
visited rooms] out of [number of rooms] room[s]."
Understand "rooms" as requesting the room tally.
```

It is important not to use "out of world" actions for anything affecting what goes on in the fictional world, or realism will collapse, and action-processing may also fail to work in the usual way. "Out of world" actions should be reserved for providing commands like ROOMS, which monitor events rather than participate in them.

Start of Chapter 12: Advanced Actions

Back to §12.14. Actions for any actor
Onward to §12.16. Reaching inside and reaching outside rules

Example 210: ★ **Spellbreaker** P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

Example 211: ★★★ **A point for never saving the game** In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

---

§12.16. Reaching inside and reaching outside rules

The flow chart back at the start of this chapter shows that, early on in processing an action (between Before and Instead), Inform asks the question "Can we see or touch things?" This is where it enforces the requirements in the action's definition:

```
Photographing is an action applying to one visible thing and requiring
```

```
light.
Scraping it with is an action applying to two things.
```

Seeing and touching are two different questions, which Inform answers in different ways. We shall see ways to modify or entirely alter what can be seen using the "deciding the scope of something" activity when we get to the Understanding and Activities chapters, and later in this chapter we will change the definition of touchability. What both have in common is that they are complicated questions, affected by the circumstances. We cannot simply declare that the player can touch a given lever, or can see in a given room: we must arrange for there to be no barriers between the player and the lever, or for there to be a light source in the room.

An example of rules applying to given objects is provided by the way that Inform decides whether the player can reach something or not. For instance, suppose the following:

```
The Laboratory is a room. In the Laboratory is a conical flask. The
flask is closed and transparent. In the flask is an antibumping
granule.
```

The player will be able to examine the granule but not to take it, as that would require reaching through glass. Suppose the player does type TAKE GRANULE: then Inform looks for potential barriers between the player and the granule, and of course finds the conical flask. If, as in this case, the thing to be touched is on the inside, then Inform asks the "reaching inside" rules for permission. There are two reaching inside rules built in to Inform:

```
can't reach inside rooms rule
can't reach inside closed containers rule
```

and in fact the second of these rules will cause the taking action to fail, because the conical flask is a closed container. (The other rule has to do with a player in one room able to see another room through, say, a telescope - merely having a line of sight doesn't give the ability to reach into the frame.)

Symmetrically, Inform also has "reaching outside" rules, used if the player is inside something and wants to reach an object in the wider room. (From a bed, probably yes; from a cage, probably no.) This ordinarily contains just one rule:

```
can't reach outside closed containers rule
```

Actions

Back to §12.15. Out of world actions

Onward to §12.17. Visible vs touchable vs carried

Example 212: ★★ **Carnivale**
An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

§12.17. Visible vs touchable vs carried

To recap, actions are created like so:

> Photographing is an action applying to one visible thing and requiring light.
> Depositing it in is an action applying to two things.
> Taking inventory is an action applying to nothing.

Actions can involve up to two different things. We can place additional requirements on any of these things by describing them as a "visible thing", "touchable thing" or "carried thing". (If we simply say "thing" or "things", as in the second example, Inform assumes the requirement to be "touchable".) These three conditions are increasingly strong:

- To be "visible", something needs only to be possible to refer to by the player, which in practice means that it must be visible to the player-character. The noun or second noun produced by any action resulting from a command at the keyboard will always satisfy this minimal condition.

- To be "touchable", the player-character must be able to physically touch the thing in question: this normally means that it must be in the same room, and there must be no physical barriers in between.

- To be "carried", the player-character must (directly) carry the thing in question. (But if the player types a command using an action requiring something "carried", like WEAR HAT, the thing in question - the hat - will sometimes be picked up automatically. This is called "implicit taking", and results in text like "(first taking the top hat)" being printed.)

If an action involves two things, they need not have the same requirement as each

other:

> Waving it at is an action applying to one carried thing and one visible thing.

Thus to "wave magic wand at banyan tree", the player must be holding the wand, but need only be able to see the tree.

Note one special case. Requirements on touchability are waived in the case of "try" actions applied to people other than the player where the things they would need to touch are doors or backdrops. (This is a compromise to avoid difficulties arising from the ambiguous locations of such items.)

Start of Chapter 12: Advanced Actions

Back to §12.16. Reaching inside and reaching outside rules

Onward to §12.18. Changing reachability

Example 213: ★★ **Eddystone** Creating new commands involving the standard compass directions.

Example 214: ★★★ **Slogar's Revenge** Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

---

§12.18. Changing reachability

The question of what the player can, and cannot, reach to touch is important in interactive fiction. It contains some of the subtlest ideas in the model world, though they often go unnoticed. For instance, if a key is on a shelf which is part of a closed box, can we reach for the key? This comes down to whether the shelf, described only as "part of" the box, is on the inside or the outside: and in fact, because it cannot know which is the case, Inform allows either. So in general it is best to regard "parts" as being exterior parts, but to avoid having parts on containers that might in the course of play be closed up with the player inside.

We can, if we wish, change the principles of what can be touched by writing new reaching inside or reaching outside rules. Returning to the example of the conical flask:

```
A rule for reaching inside the flask: say "Your hand passes through
the glass as if it were not there, chilling you to the bone."; allow
access.
```

(Or this could equally be called "a reaching inside rule for the flask".) More generally, we could give the usual flexible description of what the rule applies to:

```
A rule for reaching inside open containers: say "Your hands seem
enigmatically too large for [the container in question]."; deny access.
```

The "container in question" is the one to which the rule is being applied. Note that a reaching inside rule can "deny access" (stopping with failure), or "allow access" (stopping with success), or neither, in which case the decision is left up to any subsequent rules in the rulebook to make. If none of them decide, access is allowed.

If it seems possible that these rules will be employed by people other than the player, then we need to write them a little more carefully, and in particular we need to ensure that they print nothing for other people. In the first case below, anybody can reach through the glass; in the second case, only the player cannot reach into open containers.

```
A rule for reaching inside the flask:
    if the person reaching is the player, say "Your hand passes
through the glass as if it were not there, chilling you to the bone.";
    allow access.
```

```
A rule for reaching inside open containers:
    if the person reaching is the player:
        say "Your hands seem enigmatically too large for [the container
in question].";
        deny access.
```

The "person reaching" is, as its name suggests, the person trying to reach through the barrier in question.

Actions
Back to §12.17. Visible vs touchable vs carried
Onward to §12.19. Changing visibility

Example 215: ★ **Magneto's Revenge** Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

Example 216: ★ **Waterworld** A backdrop which the player can examine, but cannot interact with in any other way.

Example 217: ★★ **Dinner is Served** A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

---

§12.19. Changing visibility

Ordinarily, Inform has a simple model for visibility: it is either fully light or it is fully dark, and certain actions are impossible in the dark, such as examining something.

We first need to remember that darkness affects what actions are even tried, as far as the player's typed commands go. If the player is in a dark room, and there is a screwdriver on the floor, the command EXAMINE SCREWDRIVER will not try any action: the screwdriver is not "in scope", which means that the parser thinks the player does not have any means of knowing it exists. (The rules for scope can be modified - see the chapter on Activities.) But let's suppose that the player types EXAMINE BOOK, and is holding the book in question. The book is now "in scope", so the action "examining the book" is tried.

Some actions require light to be present, and "examining" is one of those. So Inform consults the visibility rules to see if it can go ahead. By default, there is only one visibility rule, which says "yes" in the light and "no" in darkness. Here, though, we create another one:

```
Visibility rule when in darkness:
    if examining the book:
        say "You have to squint. Still...";
        there is sufficient light;
    there is insufficient light.
```

A visibility rule must always conclude "there is sufficient light", or "there is insufficient light", or else do nothing and leave it to other rules to decide.

It is a possibly unexpected fact that "looking" does not require light, but instead behaves differently in darkness - it prints a pseudo-room-description such as

```
Darkness
It is pitch dark, and you can't see a thing.
```

instead of printing the description of the player's current room. This means that the "looking" action is unaffected by visibility rules. All the same, what "looking" does in the dark can be changed by using the two activities "printing the name of a dark room" and "printing the description of a dark room" (see the Activities chapter for details).

Start of Chapter 12: Advanced Actions
Back to §12.18. Changing reachability
Onward to §12.20. Stored actions

Example 218: **Flashlight**
Visibility set so that looking under objects produces no result unless the player has a light source to shine there (regardless of the light level of the room).

---

§12.20. Stored actions

As we have seen, to describe an action fully takes a complicated little bundle of information - we need to know what is to be done, who will do it, and what it will be done to. There are times when we would like to remember an action and look back on it later (perhaps many turns later, after many other actions have taken effect) - but

this is not easy to do with only the techniques we have seen so far. There are quite a few cases to get right, and it would be easy to not store quite enough of the details.

Fortunately, Inform provides a kind of value called "action" which can do all of this automatically. (In older versions of Inform this was called "stored action", but the word "stored" is now unnecessary, and makes no difference.) As with most other kinds of value, actions can be held in variables, "let" values, properties or table columns. For example:

```
The best idea yet is an action that varies.
```

creates a variable called "the best idea yet" which holds an action.

This will normally be created holding the default value - the player waiting. We really only have two ways to make more interesting actions. One is by typing them out explicitly, like so:

```
now the best idea yet is pushing the button;
```

Here "pushing the button" is a constant of the kind "action", so it goes into happily into "best idea yet" in the same way that a number like 3 could go into a number that varies. The action must be specific in every respect, so "taking something" or "doing something" will not work - "taking something" is really a general description of many possible actions, not an action in its own right.

The other way to produce a useful action is:

current action ... action

This phrase produces the action currently being processed as a value - it literally stores the action, and remembers, if necessary, the exact wording of the player's command at the time it was stored - so that even actions arising from commands like LOOK UP X100 IN THE CODE BOOK can be stored faithfully. Examples:

```
let the present whim be the current action;
say "How you would like to be [current action].";
```

This only makes sense if an action is currently going on, so it shouldn't be used in "every turn" rules, for instance.

So much for making actions: now for making use of them. The first obvious idea is to store up an action for several turns and then have it take effect later. That's easily done: just as we can "try" any action written out explicitly, so we can also try a stored one. The phrase to do this has exactly the same wording either way, since it does the same thing either way.

But actions can still be useful even if we never intend to try them. For one thing, we can say them, and this produces a fairly natural description of what the action is:

```
Before doing something in the presence of the bearded psychiatrist:
say "'Zo, the subject vishes to engage in [the current action]. Zis is
very interesting.'"
```

will produce text such as:

```
"So, the subject vishes to engage in rubbing the fireman's pole. Zis is
very interesting."
```

One of Inform's most convenient features is its ability to test if the action being processed matches vague or complicated descriptions of whole classes of actions. For example,

```
if the best idea yet is taking something, ...
```

works even though "taking something" is not a single action; it's a description which could apply to many different actions (taking a box, taking a ball, and so on). What Inform tests is whether the "best idea yet" value, a single action, fits this description or not. We can be even vaguer:

```
if the best idea yet is doing something to the lever, ...
```

Just occasionally, this can lead to ambiguities. For instance,

```
if the current action is wearing something, ...
```

fails because Inform thinks "wearing" is meant in the sense of the current action having clothes on, so it produces a problem message. To avoid this, simply write:

```
if the current action is trying wearing something, ...
```

which can't be misunderstood. Something else to be aware of is that the terms "actor", "noun" and so on will refer to that action: for instance, in

```
if the best idea yet is taking the noun, ...
```

"noun" here refers to the noun in "best idea yet", not to its meaning outside of this phrase (if indeed it has such a meaning).

When dealing with actions, we sometimes want to know what they are dealing with. We can extract this information using the following phrases:

action name part of (action) **... action name**

This phrase produces the action name part of an action. Example: suppose the current actor is Algy, who is throwing the brick at Biggles. Then

```
action name part of the current action = throwing it at action
```

noun part of (action) **... object**

This phrase produces the (first) noun of an action. Example: suppose the current actor is Algy, who is throwing the brick at Biggles. Then

```
noun part of the current action = the brick
```

If the noun is something other than an object, this produces just "nothing", the non-object.

second noun part of (action) **... object**

This phrase produces the second noun of an action. Example: suppose the current actor is Algy, who is throwing the brick at Biggles. Then

```
second noun part of the current action = Biggles
```

If the second noun is something other than an object (for instance for the command SET DIAL TO 3417 it would be the number 3417), this produces just "nothing", the non-object.

actor part of (action) **... object**

This phrase produces the person who would be carrying out the action if it were being tried. Example: suppose the current actor is Algy, who is throwing the brick at Biggles. Then

```
actor part of the current action = Algy
```

The following phrase is a convenient shorthand form:

if (action) **involves** (object)**:**

This condition is true if the object appears as any of the actor, the noun or the second noun in the action. Example:

```
if the current action involves Algy
```

would be true for "give revolver to Algy", "Algy trying flying the Sopwith Camel", "examine Algy" and so on, but false for "ask Raymond about secret airfield".

action of (an action) **... action**

This phrase is now seldom needed. It produces a literally typed action as a value. Example:

```
now the best idea yet is the action of pushing the button;
```

Nowadays in most contexts we can just type "pushing the button" as a value, and that will work fine, so this phrase is retained only to keep old code working.

Start of Chapter 12: Advanced Actions

Back to §12.19. Changing visibility

Onward to §12.21. Guidelines on how to write rules about actions

Example 219: ★ **Bosch** Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports.

Example 220: ★ **Cactus Will Outlive Us All** For every character besides the player, there is an action that will cause that character to wither right up and die.

Example 221: ★★ **Actor's Studio** A video camera that records actions performed in its presence, and plays them back with time-stamps.

Example 222: ★★ **Anteaters** The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing actions that apply to topics, as in "look up anteater colonies in the guide".

Looking at the action-processing diagram, there seem to be a bewildering number of ways to intervene. For instance, suppose it must be fatal to pick up a land mine. All six of the following rules would do the business:

```
Before taking the land mine: end the story saying "Bang!"
Instead of taking the land mine: end the story saying "Bang!"
Check taking the land mine: end the story saying "Bang!"
Carry out taking the land mine: end the story saying "Bang!"
After taking the land mine: end the story saying "Bang!"
Report taking the land mine: end the story saying "Bang!"
```

So which should we use? Of course, we could decide that it really doesn't matter: what works, works. But it is a good idea to play along with the conventions used by Inform, if only because that will make our rules interact better with each other and with rules by other people which we may someday want to borrow. So this chapter ends by offering a few guidelines. Let us suppose that we have some effect which we want to achieve.

1. Are we just trying to correct the player's typing?

For instance, responding to the command "STEAL GOLDEN EAGLE" with a reply like "To steal something, just try to TAKE it." It is bad style to make a special action for this, which does nothing except to print up this text: better is to use the "Understand ... as a mistake" technique, which will come up in the chapter on Understanding.

2. Does the effect apply only to a particular situation, or is it a general phenomenon?

In other words, does the effect apply only to particular people, things or places, or is it a generic rule of play? In the case of the land mine, this is an easy question to answer: it is a unique situation. On the other hand, stopping the player from carrying unduly heavy weights would be a generic rule of play.

Rules like the one saying that photography is only possible if one holds the camera are, by convention, also counted as generic rules of play: they are not really special rules about the camera, but apply to all possible acts of photography anywhere, so are actually generic.

Sometimes we can choose our own answer to this question, and go either way. Suppose we want a certain place to be muddy-floored, affecting things that happen there. One way would be to write exceptional rules applying to that one room. But we could alternatively create a general concept of muddiness ("A room can be clean or muddy. A room is usually clean.") and then regard the new behaviour as being a set of generic rules applying in muddy rooms. We could then, of course, create a second muddy room with much greater ease, or transplant these rules to other works and have muddy rooms in those too.

3. Particular situations: use Instead or After (or sometimes Before).

The next question is: does the effect kick in after the hoped-for action has taken place, or not? In the case of the land mine, to answer that means deciding whether we think the detonator is sensitive to the slightest touch - in which case the explosion would happen at the first touch, and should be in an "Instead" rule - or whether one must actually pick up and disturb the mine - in which case an "After" should be used.

That leaves us a choice of two rulebooks if the effect takes place when the merest impulse towards the action is felt: "Before" and "Instead". Which to use? In cases of doubt, choose "Instead". But if the effect is intended to absolutely suppress all such impulses - for instance, in a silent examination room there must be no talking - then "Before" might be more appropriate. We could imagine that someone about to say something first has a mental impulse to speak, then opens his mouth so that it becomes visible to others that some talking is about to go on, and finally utters words. Here are three possible responses:

> "You cannot contemplate breaking this smothering silence." (*Before*)
> "The invigilator stares you down through her horn-rimmed glasses." (*Instead*)
> "Everyone turns, appalled, as the silence is broken like the surface of a swimming pool by a falling elephant." (*After*)

4. Generic situations: work with action rules (or sometimes Before).

(a) If the effect takes place only when an action is definitely being tried, then we should use one of the action's three rulebooks: check, carry out or report. Check rules should do nothing, and should say nothing either unless they block the action (in which case, they should say why).

Carry out rules must not block the action - it is too late for that - and should not say anything - that hasn't happened yet. (There are a few exceptions: if the action is to look at something, then carrying it out is in a sense the same thing as reporting it. But in all cases of doubt, a carry out rule should say nothing.) Adding a carry out rule to an existing action can make it do something extra.

Report rules must neither block the action nor do anything. Working with new report rules is a way to make more natural-seeming, or more informative, messages appear. For instance, an effect where we want to be able to see through a door when it is first opened ("You open the panelled door, through which is the Board Room.") would be a case for a report rule.

In all cases, it is good style to write check, carry out or report rules in such a way that they could be used in other works too, or in situations that could conceivably have happened in this one (even if in fact it never does). We may one day want to put our new rules into an extension to be used by other people or in other projects, after all.

In this chapter, we have only seen the addition of new rules. We could add a new "check taking" rule, for instance, with the techniques seen so far. But what if the effect we want is not a matter of adding a rule but taking away, or restricting the

applicability, of an existing one? In that case, we will need to say that the rule "does nothing" under certain circumstances (see the Rulebooks chapter). The check, carry out and report rules for all of the built-in actions are named, and they appear in the Actions index.

(b) If the effect takes place to divert or supplement an action, before that action actually takes place, then this should be done with a "Before" rule. This is the biggest practical use of "Before" rules: to try other actions, either instead or as well as the one just getting under way.

For example, if we want an automatic mechanism to try opening a container before taking something inside it, that would be a classic case for "Before". Indeed, that is the only way it could work - "Before" rules have a chance to get in before the touchability conditions are tested.

If we wanted a special "stealing" action for the act of taking another person's possessions, we might want to divert any taking action for such items into our new "stealing" action - that too would be a "Before". This would ensure that any "Instead" rules to do with taking do not apply.

5. Changing the behaviour of out-of-world actions.

Remember that "Instead", "Before" and "After" do not apply: so use "check" rules to forbid certain out-of-world activities, or specify that their rules do nothing (see the Rulebooks chapter).

Finally...

These are only guidelines. The system is designed to be flexible in order to give the author the widest possible range of options, and nobody should feel ashamed of making use of them.

Start of Chapter 12: Advanced Actions

Back to §12.20. Stored actions

Onward to Chapter 13: Relations: §13.1. Sentence verbs

Examples from Chapter 12: Advanced Actions

Start of this chapter

Chapter 13: Relations

Indexes of the examples

184 **Example Virtue** *RB*

Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

```
"Virtue"

The Cloister is a room. Lady Teresa is a woman in the
Cloister. Mother Margaret is a woman in the Cloister.

Attacking someone is misbehavior. Kissing someone is
misbehavior.

Instead of asking someone to try misbehavior: say "[The
person asked] stiffens, offended by your suggestion."

Test me with "kiss margaret / margaret, kiss me".
```

---

185 **Example Latris Theon** *RB*

A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

To begin with, we create an action for going to a named place. All that this action will do is to change that person's hoped-for destination: the actual moving around comes later.

```
"Latris Theon"
```

```
A person has a room called destination.

Understand "go to [any room]" as going vaguely.

Going vaguely is an action applying to one visible thing.

Carry out someone going vaguely:
    now the destination of the person asked is the noun.

Report someone going vaguely:
    say "[The person asked] looks amused, but accepts
the commission to go to [the noun]."
```

It stands to reason the player plays Zeus or at the very least Apollo, but let's not let this go to the player's head. Note that the following rule applies to the player, but not to anyone else, so HERMES, GO TO ATHENS will work but GO TO ATHENS will not.

```
Carry out going vaguely:
    say "You're too thoroughly lost."
```

And finally we recreate Greece and one of its heroes.

```
Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.

Hermes is a man in Corinth. The destination of Hermes
is Corinth. [So he is initially happy where he is.]
Persuasion rule for asking Hermes to try going vaguely:
persuasion succeeds. [But he is open to suggestions.]
```

```
Every turn when the destination of Hermes is not the
location of Hermes:
    let the right direction be the best route from the
location of Hermes to the destination of Hermes;
    try Hermes going the right direction.
```

```
Test me with "hermes, go to athens / e".
```

It simplifies matters that our map of Greece makes it possible to reach any location from any other location, by some sequence of movements: if there were an isolated location -- say, Crete -- with no map connection to the mainland, then we would have to worry about the "right direction" not being a direction at all. The following version of Hermes' trekking rule is protected against the possibility:

```
Every turn when the destination of Hermes is not the
location of Hermes:
    let the right direction be the best route from the
location of Hermes to the destination of Hermes;
    if the right direction is a direction, try Hermes going
the right direction.
```

Example **The Hypnotist of Blois** *RB*

A hypnotist who can make people obedient and then set them free again.

```
"The Hypnotist of Blois"
```

```
A person is either hypnotized or alert. A person is
usually alert.
```

```
Persuasion rule for asking a hypnotized person to try
```

```
doing something:
    persuasion succeeds.


Understand "hypnotize [someone]" as hypnotizing.


Hypnotizing is an action applying to one thing.


Check hypnotizing:
    if the noun is hypnotized, say "Already done." instead.


Carry out hypnotizing:
    now the noun is hypnotized.


Report hypnotizing:
    say "[The noun] slips under your control."


Instead of waking someone hypnotized:
    now the noun is alert;
    say "[The noun] returns abruptly to consciousness."
```

We will learn more about check rules for other characters shortly, but the following prevents the hypnosis patient from trying to hypnotize us in turn:

```
Check someone hypnotizing someone:
    stop the action.


Maison de la Magie is a room. "In a darkened room, a
few hundreds of paces from the chateau of Blois, you
give to tourists three shows a day: displaying to them
power they do not comprehend and spectacles they do
```

not deserve."

A volunteer is a woman in the Maison. "A volunteer from the audience stands facing you, [if alert]skeptically awaiting hypnosis[otherwise]her face worshipful and obedient[end if]." The printed name of the volunteer is "volunteer from the audience". The description is "A distracted, susceptible woman." The volunteer wears a t-shirt and a baseball hat.

The player wears a top hat and a cape.

Test me with "volunteer, remove hat / hypnotize volunteer / look / volunteer, remove hat / wake volunteer / volunteer, wear hat".

---

187 **Example** **Police State** *RB*

Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

"Police State"

Seventh Avenue is a room. "The bars are all closed now, and there aren't any good clubs to go to, so you're on your own for open-air entertainment."

Persuasion rule for asking the policeman to try doing something: persuasion fails.

```
Persuasion rule for asking someone to try doing
something: persuasion succeeds.
```

Note that the policeman will never get to the second persuasion rule, so he will always refuse to do the player's nefarious bidding.

```
Charles, Thomas, and Larry are men in Seventh
Avenue. Patricia is a woman in Seventh Avenue.
```

And here's an unnecessary aesthetic touch from a later chapter, which will round up the descriptions of your friends into a single paragraph:

```
Rule for writing a paragraph about someone who is not
the policeman:
    let X be the number of visible people who are not the
policeman;
    say "It's just [X in words] of you now: [a list of visible
people who are not the policeman]. But it sure has been
a rip-roaring evening."
```

```
The policeman is a man in Seventh Avenue. "A
policeman with a very guarded expression is watching
you."
```

```
Singing is an action applying to nothing. Understand
"sing" as singing.
```

```
Report singing:
    say "A little the worse for wear, you sing."
```

```
Smelling a person is disorderly conduct. Tasting a
person is disorderly conduct. Jumping is disorderly
conduct. Singing is disorderly conduct.
```

```
Instead of someone trying disorderly conduct in the
presence of the policeman:
    say "The policeman arrests [the person asked]!";
    now the person asked is nowhere;
    the rule succeeds.


Instead of disorderly conduct in the presence of the
policeman:
    end the story saying "The policeman arrests you!"


Test me with "charles, look / charles, jump / look /
policeman, sing / thomas, taste policeman / patricia,
sing / look / jump".
```

Notice the difference between the two rules about disorderly conduct: the one for other people says 'the rule succeeds' to make sure that the action is counted as a success and not (as normally happens with instead rules) a failure. Most of the time we don't care whether actions are judged successes or failures, but it matters here, because if we type CHARLES, JUMP and the result fails, then text such as 'Charles is unable to.' will be printed - which would get in the way. So we declare the action a success.

ExampleGeneration X *RB*

A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

```
"Generation X"


The Volunteer Center is a room. "A fairly spartan office,
though there are a few attractive posters from
advertising campaigns of the past, and an ominous map
charting the deforestation of Brazil."


The desk is scenery in the Volunteer Center. "Your
```

standard metal desk, with a drawer for postal supplies and the like." On the desk is a pile of leaflets and a pile of business cards. The power stapler is a device on the desk. A drawer is part of the desk. It is openable and closed. In the drawer are a sponge and a roll of bulk-mail stamps.

Instead of doing something:
    if examining, continue the action;
    if searching, continue the action;
    if looking, continue the action;
    if asking Jenna to try doing something, continue the action;
    say "The whole point of this exercise is to train Jenna. Once you have her on the envelope-stuffing, you can go make some calls."

Instead of answering Jenna that something:
    say "You're going to have to break the instructions down to the simplest ones possible, given that she's in the mood to misunderstand on purpose."

A cardboard box is in the Center. In the cardboard box is a pile of empty envelopes. The box is openable and closed. The description is "A recycling symbol is prominent on the side, which makes you feel a little bit better about using a mailing campaign for this cause."

Jenna is a woman in the Volunteer Center. "Your daughter Jenna is here, barely visible through the cloud of resentment." The description is "Jenna [if jenna carries something]has [the list of things carried by Jenna][otherwise]returns your stare, with added hostility[end if]."

A persuasion rule for asking Jenna to try doing something: persuasion succeeds.

Unsuccessful attempt by Jenna doing something:
    repeat through Table of Retorts:
        if the reason the action failed is the cause entry:
            say "[response entry][paragraph break]";
            rule succeeds;
    say "'I don't see how I'm supposed to do that, Mom,' Jenna says."

Table of Retorts

cause	response
can't take yourself rule	"'Is that like, yoga?' Jenna asks, momentarily diverted."
[an unlikely impossibility]	[an unlikely impossibility]
[an unlikely impossibility]	[an unlikely impossibility]
[an untaken item]	[an untaken item]
[an untaken item]	[an untaken item]
[an undropped item]	[an undropped item]
[an undropped item]	[an undropped item]
[an unworn item]	[an unworn item]
[an unworn item]	[an unworn item]
[an unclosed item]	[an unclosed item]
[an unclosed item]	[an unclosed item]
[an unopen item]	[an unopen item]
[an unopen item]	[an unopen item]
[an unswitched off item]	[an unswitched off item]
[an unswitched on item]	[an unswitched on item]
[an unlocked item]	[an unlocked item]
[an unlocked item]	[an unlocked item]

To say physical impossibility:
    say "'Maybe you should've brought someone a little stronger,' Jenna says. 'Like the Incredible Hulk.' "

To say already done:
    repeat through Table of Bored Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "'Okay, I'm going to be, like, in the car,' says Jenna. 'Outside.' ";
    end the story saying "Jenna has gotten fed up".

Table of Bored Remarks

response
"'Did that,' says Jenna."
"'Check,' says Jenna."
"'Yeah, Mom, I already did that, okay?'"
"'Look, if I have to be here doing dumb stuff, could you at least tell me to do stuff that isn't already done?' Jenna asks wearily."
"Jenna gives a great upward sigh, riffling her bangs. 'Once again... there is totally no point.'"

Test me with "e / x jenna / jenna, get stapler / jenna, get stapler / x jenna / jenna, drop stapler / jenna, drop stapler / jenna, open box / jenna, open box / jenna, switch stapler on / jenna, switch stapler on / jenna, take desk / jenna, open box / jenna, open box".

ExampleIQ Test

RB

Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

We have already seen how Before... rules can generate implicit actions for the player, so that the player will, for instance, open doors before trying to walk through them. The same can be done for characters other than the player; so for instance:

"IQ Test"

The Donut Shop is a room. "Vibrantly decorated in donut colors: pink, brown, and cream."

Ogg is a man in the Donut Shop. "Ogg is slumped in the corner[if Ogg carries something] with [a list of things carried by Ogg][end if]. He wears a nametag which says 'HELLO MY NAME IS OG.'" Understand "og" as Ogg. Ogg wears a nametag. The description of the nametag is "Sadly misspelled."

```
The Donut Shop contains a transparent closed
openable locked lockable container called a case. The
case contains some cake donuts. The donuts are
edible.


The matching key of the case is a silver key. The silver
key is carried by Ogg.


A persuasion rule for asking someone to try doing
something:
    persuasion succeeds.


Before someone opening a locked thing (called the
sealed chest):
    if the person asked is carrying the matching key of
the sealed chest, try the person asked unlocking the
sealed chest with the matching key of the sealed chest;
    if the sealed chest is locked, stop the action.


Before someone taking something which is in a closed
container (called the shut chest):
    try the person asked opening the shut chest;
    if the shut chest is closed, stop the action.


The block giving rule is not listed in the check giving it to
rules.


Test me with "open case / get donuts / og, get donuts /
og, give me the donuts / eat donuts".
```

(A more detailed set of before rules for people using doors and locks is provided in Emily Short's Locksmith extension.)

A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

"Boston Cream"

Use scoring.

The Donut Shop is a room. "Vibrantly decorated in donut colors: pink, brown, and cream."

Ogg is a man in the Donut Shop. "Ogg is slumped in the corner[if Ogg carries something] with [a list of things carried by Ogg][end if]. He wears a nametag which says 'HELLO MY NAME IS OG.'" Understand "og" as Ogg. Ogg wears a nametag. The description of the nametag is "Very neatly written."

The Donut Shop contains a transparent closed openable locked lockable enterable container called a case. The case is fixed in place. The case contains some cake donuts, some jelly donuts, and some apple fritters. "The [if unopenable]damaged[otherwise]gleaming[end if] donut case [if something is in the case]contains [a list of things in the case][otherwise]has been stripped of its contents[end if]." The cake donuts, the jelly donuts, and the apple fritters are edible.

The matching key of the case is a silver key. The silver key is in a mesh basket. The mesh basket is closed, transparent, and openable. It is in the Donut Shop.

```
Before someone taking something which is carried by
the player:
    if the person asked cannot touch the player:
        say "Ogg looks with a fixed frown at [the noun].";
        continue the action;
    say "[The person asked] comes up and shakes your
arm until you drop [the noun].";
    say line break;
    silently try dropping the noun;
    stop the action.


Before someone unlocking a locked thing with
something which is not carried by the person asked:
    try the person asked taking the second noun;
    stop the action.


Before someone opening a locked thing (called the
sealed chest):
    if the person asked can see the matching key of the
sealed chest:
        if the matching key of the sealed chest is enclosed
by the sealed chest:
            say "Seeing the paradoxical location of [the
matching key of the sealed chest], [the person asked]
gives a howl of rage.";
            increment the score;
            end the story finally saying "You have thwarted
Ogg";
        otherwise:
            try the person asked unlocking the sealed chest
with the matching key of the sealed chest;
        stop the action.


Before someone taking something which is in a closed
container (called the shut chest):
    try the person asked opening the shut chest;
    stop the action.


Ogg has a number called hunger. The hunger of Ogg is
```

```
Every turn:
    increment the hunger of Ogg;
    if the hunger of Ogg is 2 and Ogg is visible, say
"Ogg's stomach growls.";
    if the hunger of Ogg is 3 and Ogg is visible:
        if Ogg can see an edible thing (called the target):
            say "Ogg eyes [the target] with obvious intent.";
        otherwise:
            say "Ogg glances at you in a disturbingly shifty
way.";
    if the hunger of Ogg is greater than 3:
        if Ogg carries an edible thing (called the target):
            try Ogg eating the target;
        otherwise:
            let new target be a random edible thing which
can be seen by Ogg;
            if the new target is a thing:
                try Ogg taking the new target;
            otherwise:
                if Ogg can touch the player, end the story
saying "Ogg is gnawing your ankle";
                otherwise try Ogg taking the player.


The crumbs are a thing. "Crumbs of [the list of edible
things which cannot be seen by the player] lie scattered
over the whole floor."


Instead of asking Ogg to try doing something when Ogg
cannot touch the player:
    say "Ogg tilts his head and shrugs, unable to hear
your instruction clearly."


Instead of asking Ogg to try eating something:
    say "It's not as though Ogg really needs any
encouragement in that department, is it?"


Definition: Ogg is hungry if the hunger of Ogg is greater
```

than 2.

Persuasion rule for asking Ogg to try doing something: if Ogg is hungry, persuasion fails; persuasion succeeds.

Persuasion rule for asking Ogg to try giving something edible to the player:
    persuasion fails.

Unsuccessful attempt by Ogg doing something:
    if the reason the action failed is a failing listed in the Table of Ogg Retorts:
        say "[reply entry][paragraph break]";
    otherwise:
        say "Ogg looks adorably confused."

Table of Ogg Retorts

failing	reply
can't take people's possessions rule	"Ogg take people!"
can't take things unheavy rule	"Ogg take that, needy!"
can't take scenery rule	"Ogg take scenery!"
can't take what's fixed in place rule	"Ogg take what's heavy."
can't drop what's not held rule	"Ogg take, then drop?"

Carry out Ogg eating an edible thing:
    move the crumbs to the holder of Ogg;
    now the hunger of Ogg is 0.

Report Ogg unlocking something with something:
    say "Ogg struggles a bit with [the second noun] in the lock of [the noun], but does eventually succeed. 'Hunh!' says Ogg.";
    stop the action.

```
Carry out Ogg opening the case when the case has
been open:
    now the case is unopenable.


Instead of closing the unopenable case:
    say "The glass panels are no longer properly seated
in their tracks, and the case cannot be closed ever
again."


Report Ogg opening the unopenable case:
    say "Ogg forces [the case] so hard that it does
break.";
    stop the action.


Report Ogg opening the case:
    say "Ogg yanks [the noun] open with such force that
you fear for its structural integrity.";
    stop the action.


Report Ogg taking something edible:
    say "Ogg acquires [the noun] with a look of tender
affection.";
    stop the action.


Report Ogg eating something:
    say "Ogg chows down on [the noun], scattering
crumbs in all directions.";
    stop the action.


Report Ogg eating something when the number of
visible edible things is 1:
    say "Ogg eats [the noun] in his trademark style. You
can no longer bear to watch.";
    stop the action.
```

After entering the case:
    say "You climb inside the case, folding yourself up uncomfortably."

After locking the case with something when the player is in the case:
    say "You turn the key firmly in the lock -- amazing it locks from within, but it does -- and settle yourself for a long wait, hoping this thing is not air-tight."

The maximum score is 1.

Test me with "open mesh / get key / unlock case / open case / enter case / close case / lock case / wait / wait".

ExampleRed Cross *RB*

A DIAGNOSE command which allows the player to check on the health of someone.

"Red Cross"

Diagnosing is an action applying to one visible thing.
Understand "diagnose [something]" as diagnosing.

Check diagnosing:
    if the noun is not a person, say "Only people can have diagnoses." instead.

```
Carry out diagnosing:
    say "You're not a medical expert."


Instead of diagnosing the player:
    say "You feel fine!"


The Clinic is a room. Fred is a man in the Clinic. The
player carries a tongue-depressor.


Test me with "diagnose me / diagnose fred / diagnose
tongue-depressor".
```

---

192 **ExampleFrizz** *RB*

Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

Suppose we have some items that get wet in contact with other damp things; in particular, if we touch anything wet while wearing a pair of gloves, the gloves too get damp. This requires that we be systematic about detecting all cases where contact occurs. So:

```
"Frizz"


A thing can be waterproof or porous. A container is
usually waterproof. An animal is usually waterproof.


A thing can be sodden or dry.


A container can be waterfilled or empty.
```

```
Before printing the name of a sodden thing: say "sodden
". Understand the sodden property as describing a
thing.


The player wears a pair of woolly gloves. Instead of
wearing a sodden thing: say "You dubiously
contemplate [the noun], and decide it's best to wear dry
clothing."


Every turn:
    follow the liquid distribution rules.


The liquid distribution rules is a rulebook.


A liquid distribution rule:
    repeat with item running through containers:
        if the item is open and the item is empty and the
item is in a waterfilled container:
            now the item is waterfilled;
            if the player can see the item, say "[The item]
fills up with water, of course.".


A liquid distribution rule:
    repeat with item running through things in a
waterfilled container:
        if the item is porous and the item is dry:
            if the player can see the item, say "[The item]
soaks through.";
            now the item is sodden.


A liquid distribution rule:
    repeat with item running through sodden things in the
airing cupboard:
        if the item is not The Last Man, now the item is dry.
```

The last liquid distribution rule:
    if the player carries a dry copy of The Last Man, end the story finally.

Instead of examining a waterfilled container:
    say "[The noun] is full of water[if the noun contains something visible]; it also contains [the list of things in the noun][end if]."

Instead of examining a container:
    say "Dry inside[if the noun contains something visible], and containing [a list of things in the noun]."

A book is a kind of thing. Instead of examining a sodden book, say "[The noun] is too soaked to read, thanks to someone's carelessness." Understand "book" as a book.

The copy of The Last Man is a book. The description of The Last Man is "Mary Shelley's very own original copy, loaned to you under the strictest of agreements[if the pond encloses the copy]. How it came to be in its present position is a long story, and not important at the moment: the critical thing is not to blame oneself - who could have anticipated the cricket bat, anyway? - but to fix it immediately, before anything worse occurs[end if]." The The Last Man is in the ziploc bag. The ziploc bag is waterproof and empty. It is openable, transparent, and closed.

The Back Garden is a room. The Back Garden contains a pond. The pond is a waterfilled container. In the pond is the ziploc bag. A carp is in the pond. The carp is an animal. Instead of taking the carp, say "You're not fast enough."

Rule for writing a paragraph about the pond:
    if the pond encloses the Last Man:
        say "It is a beautiful day just at the end of spring and beginning of summer. The sun shines, the trees blossom, the world conspires in Edenic cheerfulness. You can take in none of it. Your eye is on [the Last Man].";
    otherwise:
        say "The pond[if something is in the pond] (containing [the list of things in the pond])[end if] remains a figure of menace in an otherwise lushly verdant landscape."

Before printing the name of the Last Man when the Last Man is enclosed by the Pond: if the Last Man is dry, say "delicate, valuable ".

After printing the name of something (called the target) while writing a paragraph about something:
    if the Last Man is enclosed by the Pond:
        if the target is in something (called the parent), say " - which is in [the parent]".

The Kitchen is west of the Back Garden. "Not in any sense your province: Mrs Peaswell gets agitated if anyone besides herself so much as boils a kettle. In general it is best to sneak through, disturbing as little as possible and preserving the cosy domestic fiction that academics cannot cook.

The nearby stairwell leads up to the second floor."

The Stairwell is above the Kitchen. "Halfway up and halfway down." The airing cupboard is a container in the Stairwell. "An airing cupboard here contains the heating system, and is the ideal place for restoring wet items to a dry state." The airing cupboard is fixed in place.

Now: whenever the player definitely touches something, we want to follow certain rules about the transfer of liquid. These rules need to come after an action's check rules (to make sure the action really occurs) and before the carry out rules (so that nothing has moved or changed yet). So we'll borrow from the chapter on rulebooks to create a whole new stage to the action, occurring between the check and the carry out phases:

```
The post-check rules are a rulebook.


This is the post-check stage rule:
    abide by the post-check rules.


The post-check stage rule is listed before the carry out
stage rule in the specific action-processing rules.


A post-check rule (this is the dry glove rule):
    if we get wet:
        if the player wears the gloves and the gloves are
dry:
            now the gloves are sodden;
            say "(soaking your gloves in the process)";
    continue the action.


A post-check rule (this is the wet glove rule):
    if the player wears the sodden gloves:
        if the The Last Man must be touched and the Last
Man is not sodden:
            say "(soaking the parched pages of The Last
Man with the rude touch of your sodden gloves)";
            now the The Last Man is sodden;
        continue the action.


Before doing something when the player does not wear
the gloves:
    if The Last Man must be touched:
        if The Last Man is dry, say "[The The Last Man] is
```

too precious to endanger when you are not wearing gloves." instead;
    otherwise say "You hesitate instinctively, then recollect that you can hardly harm [The The Last Man] any more than it has already been harmed...".

To decide whether we get wet:
    if the noun is not a thing, no;
    if the noun dampens us, yes;
    if the second noun is not a thing, no;
    if the second noun dampens us, yes;
    no.

To decide whether (item - a thing) dampens us:
    if the item is not liquiferous, no;
    if item must be touched, yes;
    no.

To decide whether (item - a thing) must be touched:
    if the item is the noun and the action requires a touchable noun, yes;
    if the item is the second noun and the action requires a touchable second noun, yes;
    no.

Definition: a thing is liquiferous:
    if it is sodden, yes;
    if it is in a waterfilled container, yes;
    no.

Test me with "x book / x bag / get bag / take off gloves / w / u / put gloves in cupboard / z / open bag / touch book / push book / turn book / get gloves / wear gloves / get book".

Test disaster with "open bag".

Test mishandling with "get bag / open bag / get book".

Example3 AM *RB*

A shake command which agitates soda and makes items thump around in boxes.

"3 AM"

Understand "shake [something preferably held]" as shaking.

Shaking is an action applying to one carried thing.

Carry out shaking:
    say "Nothing results of your shaking [the noun]."

Instead of shaking a closed container when something is in the noun:
    say "Something rattles inside [the noun]."

Instead of shaking a closed transparent container when something is in the noun:
    say "Inside [the noun] there are banging noises produced by [the list of things contained by the noun]."

Instead of shaking an open container which contains something:

```
    say "[The list of things contained by the noun] might
fly out."
```

```
The Wawa is a room. "A convenience store, if you like
to call it that, vending the usual assortment of chips,
donuts, soda, and beer. There is something of a line at
the sandwich counter."
```

```
The box of enrobed cakes is in the Wawa. "A box of
Tastykake Enrobed Cakes has fallen off its shelf." The
description is "'Enrobed Cakes' is a fancy term for
'strange sponge-like baked good, covered in a thin shell
of waxy chocolate'. They are addictive, but not in a way
that lets you respect yourself in the morning." The box is
a closed openable container. In the box is a cake.
```

```
Instead of opening the box, say "The Wawa clerks frown
on the consumption of unpurchased foodstuffs."
```

```
The can of root beer is a closed openable container
carried by the player. The can of root beer is either
agitated or calm.
```

Because the can of root beer should have some reactions to having been shaken later in the game, we need to borrow a few ideas from the chapter on Time:

```
Instead of shaking the can of root beer:
    the can calms down in five turns from now;
    say "You give the can a good hard shake.";
    now the can is agitated.
```

```
Instead of listening to the can: say "It sounds [if
agitated]fizzy[otherwise]calm[end if]!"
```

```
At the time when the can calms down:
    now the can is calm.


The sticky mess is fixed in place. "There is a sticky
mess on the ground."


Instead of opening the agitated can of root beer:
    now the can of root beer is nowhere;
    now the sticky mess is in the location;
    say "You open the can and fizzing sweet soda goes
absolutely everywhere."


Instead of opening the calm can of root beer when the
can has been agitated:
    now the can of root beer is nowhere;
    say "The root beer is disappointingly flat. That's what
you get for shaking it up!"


Test me with "get box / shake box / open box / shake
box / listen to can / shake can / listen to can / wait / wait
/ wait / wait / wait / listen to can / open can".
```

---

194 **Example** **The Dark Ages Revisited** *RB*

An electric light kind of device which becomes lit when switched on and dark when switched off.

This will be explored more in subsequent examples, but one of the things we can do with carry out rules is extend the function of existing commands so that they do more, or have special effects in specific situations. For instance, suppose we want to have a class of electric light:

```
"The Dark Ages Revisited"
```

```
An electric light is a kind of device. Carry out switching
on an electric light: now the noun is lit. Carry out
switching off an electric light: now the noun is unlit.
```

This will not affect the behavior of any other devices when switched; it will also not change the way in which switching lights on and off is reported. The player will still see "You switch the sodium lamp on." or the like. In this case that is probably what we want. If we wanted a special way of describing turning on electric lights as opposed to all other devices, we could also add an after rule for the electric light class. Adding this rule to the carry out train does guarantee, though, that in no case will we manage to make the lamp lit without actually making it switched on (or vice versa).

```
The Stooped Corridor is a room. "A low, square-cut
corridor, running north to south, stooping you over."
```

```
The sodium lamp is an electric light in the Stooped
Corridor. "[if switched on]The sodium lamp squats on
the ground, burning away.[otherwise]The sodium lamp
squats heavily on the ground.[end if]". The description is
"It is a heavy-duty archaeologist's lamp, [if switched
off]currently off.[otherwise]blazing with brilliant yellow
light.[end if]"
```

```
Instead of burning the sodium lamp, try switching on the
lamp.
```

So far so easy. Since we've built the description of its light or darkness into the lamp's description, though, we may want to get rid of the "...is switched on" line that automatically follows when we look at something. For this we do need to borrow from a later chapter:

```
The examine devices rule is not listed in the carry out
examining rules.
```

```
Test me with "x lamp / switch lamp on / look / x lamp".
```

A CUT [something] WITH [something] command which acts differently on different types of objects.

Suppose we intend a game in which the player needs to cut things open on a regular basis. We'll want to check whether he has the proper tools handy, and deal graciously with commands such as CUT [something] when no tool is specified. So:

```
"Paddington"

A blade is a kind of thing.

Understand "cut [something] with [something]" as
cutting it with.

Instead of cutting something:
    if a blade (called the edge) is held by the player,
        try cutting the noun with the edge;
    otherwise say "Your fingernails are not sharp
enough."

Cutting it with is an action applying to two things.

Check cutting it with:
    if the noun is a person, say "That would hurt."
instead;
    if the second noun is not a blade, say "[The second
noun] has not got enough of a blade." instead.
```

```
Carry out cutting it with:
    increment the count of rips of the noun.


Report cutting it with:
    say "You slash [the noun] with [the second noun]."
```

We'll need a way to account for all these cuts and rips.

```
Definition: a thing is ripped if the count of rips of it > 0. A
thing has a number called the count of rips. After
examining something ripped, say "You see [the count of
rips of the noun in words] rip[s] in [the noun][if
something is in the noun], revealing [a list of things in
the noun][end if]."
```

Moreover, because open containers normally list their contents when examined but we'd prefer Paddington's to be mentioned in the ripping paragraph:

```
The examine containers rule does nothing when
examining the teddy bear.
```

So far, so good. But suppose that we'd like cutting also to make containers be permanently open and impossible to close again. We could write an "instead" rule, but that would mean that only our instead instructions would take effect, overriding the normal cutting it with rules entirely. Better would be to add a second carry out rule:

```
Carry out cutting a container with something:
    now the noun is open;
    now the noun is unopenable.
```

Now our rule will occur whenever a container is cut, but play will still go on to the reporting stage. And indeed we can add more of these, of varying degrees of specificity:

```
Carry out cutting something which is part of something
with something:
    move the noun to the player.
```

```
Carry out cutting the quilt with something:
    now the description of the quilt is "Horribly tattered."
```

For that matter, we might want to add a report rule as well, to occur after the "You slash..." rule, so that every time the player cuts something open which has contents, the contents will be listed.

```
Report cutting it with:
    if the noun is open and the noun contains something,
        say "Visible within [is-are a list of things in the
noun]."
```

This time we do not add the condition to the rule (i.e., Report cutting an open noun...) If we did, this report rule would be more specific than the general report rule, and would occur first.

```
The Safehouse is a room.
```

```
The teddy bear is a closed thing in the Safehouse. The
description is "Fluffy[if the head is part of the bear], with
an outsized head[otherwise], but headless[end if]." The
head is a closed part of the teddy bear. In the bear is a
large wad of stuffing. In the head are a small wad of
stuffing and a packet of smuggled diamonds.
```

```
The quilt is in the Safehouse. The description is "An old
but comforting quilt."
```

```
The player carries a blade called a switchblade.
```

Here is a final nicety to get rid of the "which is closed" statement on our closed unopenable teddy bear, using an "activity" rule:

```
After printing the name of a closed unopenable
container:
    omit contents in listing.
```

```
Test me with "cut quilt with bear / cut quilt with
switchblade / examine quilt / cut bear with switchblade /
again / examine bear / cut head with switchblade / get
diamonds / mourn loss of innocence".
```

ExampleDelicious, Delicious Rocks *RB*

Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

In some cases, we may want to add new stages to action processing. One possibility is a stage where we check the sanity of what the player is trying to do before executing any of the other commands; so that we avoid, for instance

```
>EAT ROCK
(first taking the rock)
That's plainly inedible.
```

Here is how we might insert such a stage in our action processing, using rulebook manipulation.

```
"Delicious, Delicious Rocks"
```

```
Section 1 - Procedure
```

```
The sanity-check rules are a rulebook.
```

```
This is the sanity-check stage rule:
    abide by the sanity-check rules.


The sanity-check stage rule is listed after the before
stage rule in the action-processing rules.


Section 2 - Scenario


Candyland is a room. The lollipop tree is an edible thing
in Candyland. The genuine rock is a thing in Candyland.


Sanity-check eating an inedible thing:
    say "Your digestion is so delicate -- you're sure [the
noun] wouldn't agree with you." instead.


Test me with "eat lollipop / eat rock".
```

Notice that now Inform does not try taking the rock before rejecting the player's attempt to eat it.

It is of course possible to get the same effect with

```
Before eating an inedible thing:
    say "Your digestion is so delicate -- you're sure [the
noun] wouldn't agree with you." instead.
```

...and in a small game with few rules, there's not much reason to add an extra stage. The ability to modify the stages of action processing becomes useful when we have a fairly large game with sophisticated modeling and want to be sure that some kinds of message (such as the sanity-check) are always handled before other things that we might be doing at the before stage (such as generating implicit actions like opening doors before going through them).

ExampleNoisemaking

RB

Creating a stage after the report stage of an action, during which other characters may observe and react.

Suppose the current sequence of action handling is not quite enough for us: we'd also like to have a stage after reporting, where other characters can react to the player character's behavior after it has already happened and been reported on screen. Having such a stage is unlike using "after", because after occurs before reports and prevents them from being printed. So, for instance, we could allow the player to do any of a range of different actions that make loud noises, and have a nervous bird that reacts to all of them by flying away afterward.

To do this, we can add a new rule into the specific action-processing rules. (For a list of these, see the Rules index.) Moving rules around and adding new ones requires syntax that we will learn in the chapter on Rulebooks, but the present example is fairly straightforward:

```
"Noisemaking"

Section 1 - Procedure

The other-player response rule is listed after the report
stage rule in the specific action-processing rules.

This is the other-player response rule:
    follow the observation rules.

The observation rules is a rulebook.

Section 2 - Scenario
```

Country Lane is a room. West of Country Lane is Outside the Farmhouse. East of Country Lane is Village Center. North of Country Lane is Open Field.

The player carries a drum.

The black crow is an animal in Country Lane.

Singing is an action applying to nothing. Understand "sing" as singing.

Report singing:
    say "You hum a little ditty."

Singing is a loud action.

Attacking the drum is a loud action.

The block attacking rule is not listed in any rulebook.

Report attacking something:
    say "THWACK!"

An observation rule for loud action in the presence of the black crow:
    let N be a random adjacent room;
    if N is a room, move the black crow to N;

```
say "The crow, startled, flies off to [N]."
```

```
Test me with "sing / g / n / hit drum".
```

TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

Suppose that we want to change the reporting of "take" so that the player is always told something like "You take the book from the shelf." or "You pick up the toy from the ground." In order to generate these reports, we will need to know where the object started, even though by the time we are printing the output, the object will have moved.

```
"Removal"
```

```
The Pharmacy is a room. A desk and a trash can are in
the Pharmacy. The pill-counter, a prescription, and a
computer are on the desk. The computer is fixed in
place. The pill-counter contains some Vicodin. The trash
can contains an empty box.
```

```
The taking action has an object called previous locale
(matched as "from").
```

The previous locale could in theory be either a thing or a room, so we make it "an object" -- that is, the most generic possible kind, to which both things and rooms belong. Now we record what the previous locale is at the beginning of each taking action:

```
Setting action variables for taking:
    now previous locale is the holder of the noun.
```

```
Report taking something from the location:
    say "You pick up [the noun] from the ground."
instead.


Report taking something:
    say "You take [the noun] from [the previous locale]."
instead.


Test me with "get all".
```

---

199 **Example** **Further Reasons Why All Poets Are Liars** *RB*

The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

We begin with the location and its fittings, and we create a kind of value which names the different internal positions we will allow.

```
"Further Reasons Why All Poets Are Liars"


Nook Obscure is a room. "Above the College kitchens,
which make a humming sound, less tuneable than bees,
but hardly less industrious, with shrill notes of sharp
command and scolding intermixed: and below Trinity's
loquacious clock, who never lets the quarters, night or
day, slip by him unproclaimed, and tells the hours twice
over with a male and female voice. In short, the kind of
rubbish room they give to a northern villager. But you
get a bed and a high shelf all of your own. And you long
to find some Romantic way to look out of the window."


The window, the shelf and the bed are scenery in the
```

```
Nook Obscure. The shelf and the bed are supporters.
The bed is enterable.
```

```
Internal position is a kind of value. The internal positions
are nowhere at all, over by the window, under the shelf
and near the bed.
```

```
The box is an enterable supporter in Nook Obscure. The
current box position is an internal position that varies.
The current box position is near the bed. "Your packing
case, stamped W. WORDSWORTH (KENDAL), is
[current box position]." Instead of taking the box, say "It
is filled with your peerless rock collection and too heavy
to lift, but could be pushed." Instead of opening the box,
say "It is securely nailed shut."
```

We create an action, "pushing it over to", for pushing a box around on the floor of a single location. (Calling this "pushing it over to" prevents clashes with the existing "pushing it to" action, which is for pushing things from one room to another.) Almost half of the text which defines the action is concerned with the two action variables, but they make the implementation of everything else so much easier that we end up writing less than if we hadn't used them.

```
Understand "push [box]" as a mistake ("You can push
the box to the window, the bed or the shelf.").
```

```
Understand "push [something] to [something]" as
pushing it over to. Pushing it over to is an action
applying to two things.
```

```
The pushing it over to action has an internal position
called the old position.
The pushing it over to action has an internal position
called the new position.
Setting action variables for pushing something over to
something:
    now the old position is the current box position;
    now the new position is nowhere at all;
```

```
    if the second noun is the window, now the new
position is over by the window;
    if the second noun is the bed, now the new position is
near the bed;
    if the second noun is the shelf, now the new position
is under the shelf.



Check pushing it over to:
    if the noun is not the box, say "That's not something
you can push." instead;
    if the player is on the bed, say "You can't reach from
here." instead;
    if the player is on the noun, say "Not while you are
standing on [the noun]." instead;
    if the new position is nowhere at all, say "You can
only push [the noun] to the window, the bed or the
shelf." instead;
    if the new position is the old position, say "The [noun]
is already [new position]." instead.
Carry out pushing it over to:
    now the current box position is the new position.
Report pushing it over to:
    say "With some effort, you shove [the noun] from [old
position] to [new position]."
```

Everything which remains simply provides a couple of puzzles to test this arrangement.

```
Euclid's Elements is on the shelf. Understand "euclid" or
"book" as the Elements.
Instead of taking something (called the item) which is on
the shelf:
    if the player is on the box and the current box position
is under the shelf, continue the action;
    say "You cannot reach [the item], which is up on the
shelf."



Instead of examining the window:
    say "This window opens rather unpromisingly onto
the chapel wall opposite, so even granted the moonlight
it is dark in here. Still, surely there's a poem here
somewhere?"
```

```
Instead of examining the window when the player is on
the bed:
    say "Just a blank patch of chapel wall."


Instead of examining the window when the player is on
the box:
    if the current box position is near the bed:
        say "Tantalisingly, you are not quite able to spy the
statue.";
    otherwise if the current box position is under the
shelf:
        say "All you can see is the antechapel wall, and the
dull silver gleam of the pealing organ.";
    otherwise:
        say "At last! You can just, standing on tiptoes on
the box right up at the window, make out the top of the
statue! Of such epiphanies are Poesy born. Let's see
now... oh yes...[paragraph break]And from my pillow,
looking forth by light[line break]Of moon or favouring
stars, I could behold[line break]The antechapel where
the statue stood[line break]Of Newton with his prism
and silent face,[line break]The marble index of a mind
for ever[line break]Voyaging through strange seas of
Thought, alone.";
        end the story finally.


Test me with "get on bed / x window / get off / x window
/get elements / get on box / x window / get elements /
push box to shelf / get off / push box to shelf / get on
box / get elements / x window / get off / push box to
window / get on box / x window".
```

---

200 **Example** **The Second Oldest Problem** *RB*

Adapting the going action so that something special can happen when going from a dark room to another dark room.

Text in this example is drawn from Will Crowther's original 1976 FORTRAN implementation of ADVENTURE, the founding work of the genre, whose source code was rediscovered by Dennis G. Jerz in 2007.

Note the capitals: the program ran on an early computer without lower case lettering. They look a little mimsy now, but picture them glowing green on an old-style cathode ray tube monitor in a darkened room late at night.

The problem alluded to is that the player is forbidden to walk between two dark rooms, so that he must always have light to see by from at least one end of any movement. Writing source text to achieve this is tricky to get right in every case, because the determination of light is hard to do. Here we interleave the necessary rules into the existing "going" action, using a new action variable to record the number of ends which are dark as experienced by the player, which might be 0, 1 or 2:

```
"THE SECOND OLDEST PROBLEM"


The going action has a number called the dark terminus
count.
Setting action variables for going:
    now the dark terminus count is 0;
    if in darkness, increment the dark terminus count.
The last carry out going rule:
    if in darkness, increment the dark terminus count;
    if the dark terminus count is 2, end the story saying
"YOU FELL INTO A PIT AND BROKE EVERY BONE IN
YOUR BODY!" instead.
```

And now three early rooms to try this out.

```
COBBLE CRAWL is a room. "YOU ARE CRAWLING
OVER COBBLES IN A LOW PASSAGE. THERE IS A
DIM LIGHT AT THE EAST END OF THE PASSAGE."


DEBRIS ROOM is west of COBBLE CRAWL. "YOU
ARE IN A DEBRIS ROOM, FILLED WITH STUFF
WASHED IN FROM THE SURFACE. A LOW WIDE
PASSAGE WITH COBBLES BECOMES PLUGGED
WITH MUD AND DEBRIS HERE,BUT AN AWKWARD
CANYON LEADS UPWARD AND WEST."
```

AWKWARD CANYON is west of DEBRIS ROOM. "YOU ARE IN AN AWKWARD SLOPING EAST/WEST CANYON."

DEBRIS ROOM and AWKWARD CANYON are dark.

Rule for printing the name of a dark room: say "DARKNESS" instead.
Rule for printing the description of a dark room: say "IT IS NOW PITCH BLACK. IF YOU PROCEED YOU WILL LIKELY FALL INTO A PIT." instead.

Test me with "w / e / w / w".

This is only the second oldest problem in the IF literature: the earliest puzzle is unlocking the steel grate which bars entrance to the cave.

---

201 **ExamplePuff of Orange Smoke** *RB*

A system in which every character has a body, which is left behind when the person dies; attempts to do something to the body are redirected to the person while the person is alive.

Suppose we want to let the player kill characters, leaving behind corpses.

"Puff of Orange Smoke"

Paraguay is a room. Bolivia is north of Paraguay. Lydia is a woman in Paraguay. "Lydia is, as usual, here." The description of Lydia is "Long, long legs and a sarcastic attitude." Instead of touching Lydia: say "'Watch it, sailor,' she snaps."

```
A body is a kind of thing. A body is a part of every
person. Instead of touching a body: say "[The noun] is
grotesquely inert."


The description of Lydia's body is "Long, long legs and
no attitude at all." The initial appearance of Lydia's body
is "Lydia's corpse is sprawled at your feet."
```

Using our "part of every person..." line, we've conveniently assigned one body per person. Since we're going to separate people from their bodies when the bodies die, though, we also want a more permanent relation that will help us keep track of which bodies used to belong to which people:

```
Spirit-possession relates one person to one body. The
verb to be owner of means the spirit-possession
relation.


When play begins:
    repeat with victim running through people:
        let the corpse be a random body which is part of
the victim;
        now the victim is owner of the corpse.
```

When Lydia is alive, we want >TOUCH LYDIA'S BODY to mean the same thing as >TOUCH LYDIA, so we use the setting action variables rules as a convenient point at which to reassign the action:

```
Setting action variables when the noun is a body which
is part of a person (called owner):
    now the noun is the owner.


Setting action variables when the second noun is a body
which is part of a person (called owner):
    now the second noun is the owner.
```

This doesn't change Inform's idea about what action is being performed; just about the object it's being performed on. The rest of the action will

now proceed as if the player had typed >TOUCH LYDIA.

Along similar lines, once Lydia is dead, we want >MOVE LYDIA to mean >MOVE LYDIA'S BODY if the body is in view:

```
Setting action variables when the noun is a dead person
and the noun is owner of a visible body (called the
mortal remains):
    now the noun is the mortal remains.
```

The trick is, though, that >MOVE LYDIA will only be understood if there is something called Lydia that the player can see and refer to, even after she's dead. There are various ways to do this, but the least painful here will be to make the deceased Lydia permanently visible, by putting her in an always-accessible backdrop. The backdrop itself will never be mentioned in the game, and we should make its name something that the player is unlikely to type casually; we don't want the player to interact with it directly. So:

```
The worldview is a privately-named backdrop. It is
everywhere. The spirit-world is a privately-named
transparent closed unopenable container. It is part of
the worldview.


Definition: a person is dead if he is in the spirit-world.
```

It's also possible that the player will type something like >X LYDIA when Lydia's corpse is not in view, so we should have an appropriate answer to that as well:

```
Before doing something to a dead person:
    say "[The noun] is dead; or had you blocked that
out?" instead.
```

Because the before rules happen after the setting action variables rules, this will only ever happen if the corpse is not visible.

Now we define the attack itself, which should discard the body, move the spirit to its eternal resting place, and describe the event to the player:

```
Instead of attacking someone:
    let the corpse be a random body which is part of the
noun;
    move the corpse to the location;
    move the noun to the spirit-world;
    say "With a single blow, you rid the world of [the
noun]."
```

And finally a trick borrowed from the chapter on understanding, so that we can refer to "Lydia's body" while Lydia is alive, but "Lydia's corpse" only after Lydia has died:

```
Understand "corpse" as a body when the item described
is not part of a person.
```

```
Test me with "x body / x lydia's body / touch lydia's body
/ x corpse / kill lydia / look / x lydia's body / x lydia's
corpse / x corpse / x lydia / touch lydia / lydia, hello / n /
x lydia / touch lydia / lydia, hello".
```

---

202. ExampleCroft *RB*

Adding special reporting and handling for objects dropped when the player is on a supporter, and special entering rules for moving from one supporter to another.

Suppose that we have a design in which the player spends lots of time on enterable supporters, and in which we want to report certain actions -- dropping things onto those supporters, or leaping from one to another -- in a new way. We might begin by adding some action variables to help us keep track of the situation:

```
"Croft"
```

```
The dropping action has an object called the container
dropped into (matched as "into").
```

The dropping action has an object called the supporter dropped onto (matched as "onto").

Rule for setting action variables for dropping:
    if the actor is in a container (called C), now the container dropped into is C;
    if the actor is on a supporter (called C), now the supporter dropped onto is C.

Report dropping a heavy thing onto a metallic thing:
    say "You drop [the noun], and [the supporter dropped onto] clangs protestingly." instead.

Report someone dropping a heavy thing onto a metallic thing:
    say "[The actor] drops [the noun] onto [the supporter dropped onto], which clangs protestingly." instead.

A thing can be heavy or light. A thing can be metallic or ordinary. A thing is usually ordinary. A thing is usually light.

The Ancient Cambodian Temple is a room. "A vast space built for ancient and forgotten rituals. The stone floor crawls with vermin. Well above the floor, and separated by some feet, are twin platforms built into the wall: the one carved of jointed wood, the other of sheets of graven bronze."

A platform is a kind of supporter. A platform is always enterable. A platform is usually scenery.

The bronze platform is a metallic platform in the Temple. Lara is a woman. She is on the bronze platform. She

```
wears safari pants and a tank top. She carries a gun
and a map. The gun is heavy.
```

```
The wood platform is an ordinary platform in the
Temple. The player is on the wood platform. The player
carries a rope, an Ancient Cambodian/English
Phrasebook, a pickaxe, and a precious idol. The idol
and the pickaxe are heavy.
```

```
Persuasion rule: persuasion succeeds.
```

```
The entering action has an object called the place left
(matched as "from").
    Check entering a platform from a platform:
        if actor is the player, say "You leap into midair to
cross the distance...";
    otherwise say "[The actor] leaps gracefully across the
distance...";
    move the actor to the holder of the noun, without
printing a room description.
```

Because this rule occurs before the "implicitly pass through other barriers rule", that rule will not occur when we move from platform to platform; we'll use our own custom rule instead.

```
Rule for setting action variables for entering:
    now the place left is the holder of the actor.
```

```
Report entering a platform from a platform:
    say "You land in a cat-like crouch on [the noun]."
instead.
```

```
Report Lara entering a platform from a platform:
    say "Lara lands soundlessly on [the noun][if the noun
supports the player] beside you[end if]." instead.
```

```
Report entering a platform from the location:
    say "You jump, catch the edge of [the noun] in your
hands, and -- exerting considerable upper-body strength
-- pull yourself up onto it." instead.


Report Lara entering a platform from a location:
    say "Lara jumps, catches the edge of [the noun], and
is standing upright on it, all in less time than it takes to
tell."


Instead of examining a person who is not the player:
    say "[The noun] carries [list of things carried by the
noun] and wears [list of things worn by the noun]."


Instead of climbing a platform, try entering the noun.


Test me with "Lara, drop map / lara, drop gun / drop idol
/ enter bronze platform / drop pickaxe / get off / climb
wood".
```

Example **The Man of Steel** *RB*

An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

It will sometimes be handy to write actions that are only available to the non-player characters and not to the player. To do this, we just define an action which has no "understand": the player will never be able to specify this command himself, but other characters can try it.

This is particularly useful for creating abstract or general actions for when we want a character to eat something, wear something, or go somewhere, but aren't too particular as to what the object is; as here, where we just want Clark to move away from the kryptonite, regardless of

direction:

"The Man of Steel"

Escaping is an action applying to nothing.

Carry out someone escaping:
    let space be the holder of the person asked;
    let place be a random room which is adjacent to the space;
    let way be the best route from the space to the place;
    try the person asked going way.

Every turn:
    if Clark Kent can see kryptonite, try Clark Kent escaping.

The Daily Planet Newsroom is a room.

Perry White's Office is west of the Newsroom. In Perry White's Office are a desk and a poster of Elvis. On the desk is a lead-lined box. The box is openable. In the box is the green kryptonite crystal.

The Supply Closet is east of the Newsroom. The Elevator Bank is north of the Newsroom.

Clark Kent is a man in the Newsroom. "Clark [if Clark can see the kryptonite]looks a bit ill[otherwise]is here, frowning as he revises his latest article[end if]."

```
Test me with "west / get box / east / close box / east /
west / north / south / west".
```

Replacing the inventory reporting rule with another which does something slightly different.

Inform has built-in commands for other people, and sometimes we may want to adjust the way these work without completely disabling and replacing the command. Suppose, for instance, that instead of

```
Kermit the Frog looks through his possessions.
```

we'd like someone taking inventory to report what he's got, thus:

```
Kermit the Frog says, "I seem to be carrying a
microphone and wearing a hat and a trenchcoat."
```

To do this, we could replace the built-in report rule with a different one.

```
"Trying Taking Manhattan"


The loud inventory rule is listed instead of the report
other people taking inventory rule in the report taking
inventory rules.


This is the loud inventory rule:
    unless the player is the person asked:
        say "[The person asked] says, 'I seem to be
carrying [a list of things carried by the person asked][if
the person asked is wearing something] and wearing [a
list of things worn by the person asked][end if].'"
```

Persuasion rule for asking someone to try doing something: persuasion succeeds.

Grand Central Station is a room. "Here you are in New York, New York. Any minute now someone is going to burst into song."

Kermit the Frog is a man in Grand Central Station. "Kermit the Frog stands nearby, enjoying being green." Kermit is wearing a hat and a trenchcoat. He is carrying a microphone.

Test me with "inventory / kermit, inventory".

---

205 **ExampleUnder Contract** *RB*

Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

"Under Contract"

The Sound Stage is a room. The description is "A somewhat creaky and unconvincing reproduction of the docks of San Francisco."

Clark Gable is a man in the Sound Stage. "Clark is here, carrying [the list of things carried by Clark][if Clark wears anything] and wearing [the list of things worn by Clark][end if]." Clark carries a swagger stick. Clark carries an open openable player's holdall called a briefcase. The

briefcase contains a signed contract. Clark wears a fedora and a pair of pants. A handle is part of the briefcase.

The player carries an open openable player's holdall called a frilly bag. The carrying capacity of the player is 2. The player wears a sparkly scarf and a slinky dress. A strap is part of the bag. The tent is a portable enterable container.

Instead of taking off the slinky dress: say "Rowr! Not that kind of movie, babe."

Instead of taking off the pants: say "That's not your job, whatever the society pages might suggest."

Unsuccessful attempt by Clark doing something:
    repeat through table of Clark Retorts:
        if the reason the action failed is the cause entry:
            say "[response entry][paragraph break]";
            rule succeeds;
    say "'I don't think that's in the script,' says Clark dubiously."

Table of Clark Retorts

cause	response
[try taking yourself]	"I'm already dressed,' Clark remarks. You've heard that line before, but it sounds so much more convincing from him."
[try taking other worn clothes]	"I don't think I would be appreciated if I tried to do that to [the noun],' he rumbles."
[try taking the pants]	"I don't want to open that can of worms,' Clark remarks."
[try taking people's possessions]	"I don't take people's possessions unless I'm invited,' Clark replies. 'It ain't proper.'"
[try taking someone inside]	"Do you take me for a common bellhop?' Clark demands."
[try taking what's already taken]	"I'm already what I am, darling.'
[try taking scenery]	"I'm not the scenery male, darling,' he says with a wry twinkle."
[try taking what's fixed in place]	"I'm not the what's fixed in place, darling,' he says with a wry twinkle."
[try taking too many things]	"Clark grins. 'I've only got so many hands, darling,' he says."
[try taking impossible things]	"I'm not impossible.'
[try inserting into containers rule	

[an impossible]"
[an impossible]"
[an impossible]"
[an impossible]"
"'We drop ourselves, me and me,' Clark replies, with a smile."
[already done]"
"And drop what's not [the current run],' replies Clark."
[already done]"
"[already done]"
"'Oh, I'm sorry, I'm so clumsy,' says Clark. Oh, he's so modest."
"'The second what's not a supporter thing,' says Clark reprovingly."
[already done]"
[already done]"
[already done]"
[already done]"
"'Or wearing just [the current clothes] every year,' says Clark. 'In short: no.'"
[already done]"
"'What, you mean I'm not, poison me?'"
[already done]"
[already done]"
[already done]"
[already done]"
[already done]"
[already done]"
[already done]"

To say already done:
    repeat through Table of Clark's Bored Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "'Already done.' "

Table of Clark's Bored Remarks
response
"'I have anticipated your every demand.'"
"'We've been through this part of the script already,' Clark remarks, with just a hint of creeping tedium."
"'Right right, already taken care of...'"
"'I'm a believer in rehearsal,' Clark says, sounding more bored than ever, 'but we really have got all this done already.'"

```
To say salacious retort:
    repeat through Table of Clark's Flirtatious Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "Clark, driven past the point of endurance, seizes
you in his arms and kisses you.";
    end the story finally.



Table of Clark's Flirtatious Remarks
response
"Clark allows his mustache to quirk at the suggestion."
"Clark wiggles his eyebrows at you."
"'That kind of thing isn't in my contract, sweetie,' says
Clark. 'If you're interested you'll have to ask off-
camera.'"
"'I'd feel so... bare,' Clark says, with a sidelong look at
you."
"'You first,' says Clark."
"Clark's stare is intense. You may be on the verge of
breaking him."



To say physical impossibility:
    repeat through Table of Clark's Frustrated Denials:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "Clark makes a helpless gesture about his ability
to move himself, or parts of himself, through solid
objects. "



Table of Clark's Frustrated Denials
response
"Clark frowns. 'I can't reach through things -- unless you
were planning a special effect?'"
"'Look,' says Clark. 'If you want a stage magician, hire
one.'"
"Clark just groans."
"Clark says drily, 'Is this in the script, or are we doing
improvisation now?'"
```

The block giving rule is not listed in the check giving it to rules. The block showing rule is not listed in the check showing it to rules. The report smelling rule is not listed in the report smelling rules. The report listening rule is not listed in the report listening to rules.

Carry out listening to something:
    do nothing.

Report listening to something: say "Your attention bears no interesting result."

Carry out smelling something:
    do nothing.

Report smelling something: say "Your attention bears no interesting result."

Report someone listening to something: say "[The person asked] concentrates, listening."

Report someone smelling something: say "[The person asked] sniffs at [the noun]."

Instead of asking someone for something: try asking the noun to try giving the second noun to the player.

Carry out showing something to someone: say "You reveal [the noun] to [the second noun]."

```
Carry out Clark showing something to someone:
    if the second noun is the player:
        say "Clark shows you [the noun]. [run paragraph
on]";
        try examining the noun;
    otherwise:
        say "Clark reveals [the noun] to [the second noun]."
```

```
Instead of asking someone to try saying yes: try saying
yes. Instead of asking someone to try saying no: try
saying no. Instead of asking someone to try saying
sorry, try saying sorry.
```

And now we make Clark a very amenable type:

```
A persuasion rule for asking people to try doing
something: persuasion succeeds.
```

There are quite a few actions for which no automatic other-character behavior is provided at all, because they always end in failure when the player tries them. Currently the response is not very interesting. Let's spice it up a bit:

```
Burning something is useless action. Waking up is
useless action. Thinking is useless action. Cutting is
useless action. Jumping is useless action. Tying
something to something is useless action. Drinking
something is useless action. Swinging is useless action.
Rubbing is useless action. Setting something to
something is useless action. Waving hands is useless
action. Buying is useless action. Climbing is useless
action. Sleeping is useless action. Kissing is useless
action. Throwing something at something is useless
action. Attacking is useless action. Asking something
about something is useless action. Telling something
about something is useless action. Answering
something that something is useless action. Waking
something is useless action.
```

```
A persuasion rule for asking people to try useless
action:
```

```
    say "Clark Gable frowns. 'I don't mean to be difficult,
but I can't see any point in that.'";
    persuasion fails.


A persuasion rule for asking Clark to try taking off the
pants:
    say "He gives you a look of mocking exasperation.
Apparently you'll have to take a (slightly) more subtle
approach.";
    persuasion fails.


    Check someone giving something to someone (this is
the can't give worn items rule):
        if the person asked wears the noun, stop the
action.
```

And because we do not want Clark automatically taking off the pants as the result of a drop action:

```
Instead of Clark taking off the pants: do nothing.
```

We know that this can only come about as a default action, because we have arranged matters so that he cannot be persuaded directly.

```
Test me with "give dress to clark / clark, take off pants /
clark, eat pants / clark, give me the pants / clark, drop
pants / clark, put the pants in the briefcase / again / g / g
/ g"
```

ExampleGet Axe *RB*

Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

We could now re-write the check rules so that any time someone (the player or someone else) tries to pick up a container which he is in, he will first get out:

```
"GET AXE"


This is the clever can't take what you're inside rule:
    if the person asked is in the noun, try the person
asked exiting;
    if the person asked is in the noun, rule fails.


The clever can't take what you're inside rule is listed
instead of the can't take what you're inside rule in the
check taking rules.


Attic is a room. The unused coffin is in the Attic. The
coffin is enterable and openable and open. Raskolnikov
is a man in the coffin.


Persuasion rule for asking Raskolnikov to try doing
something:
    persuasion succeeds.


Test me with "raskolnikov, get coffin".
```

ExampleBarter Barter *RB*

Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

By default, if we make no modifications, telling one player to give something to another will fail, even if persuasion succeeds. This is because the default behavior of the GIVE command is interrupted by the "block giving rule" -- since in many cases we do not want people to exchange objects freely.

However, suppose that we do want characters to be able to exchange articles freely: we allow persuasion to succeed and turn off the "block giving rule".

```
"Barter Barter"


The block giving rule is not listed in the check giving it to
rules.


A persuasion rule for asking people to try giving:
persuasion succeeds.


The Trading Post is a room.


Meriwether Lewis is a man in the Trading Post. He
carries a fluffy handmade quilt and a bag of beans. The
beans are edible.


William Clark is a man in the Trading Post. He carries
leather slippers, a journal, and a loaf of bread. The
bread is edible. The slippers are wearable.


Instead of examining someone:
    say "[The noun] is carrying [the list of things carried
by the noun]."
```

And now we might want to implement a way to keep track of whether the recipient character wants what's being offered:

```
Check someone trying giving something to someone
(this is the sneering refusal rule):
    if the second noun dislikes the noun, stop the action.
```

```
Unsuccessful attempt by someone trying doing
something:
    if the reason the action failed is the sneering refusal
rule, say "'Would you care for [the noun]?' [the person
asked] asks solicitously of [the second noun].


But [the second noun] refuses [the noun] disdainfully.";
    otherwise say "[The person asked] just appears
bewildered by this improbable instruction."


Distaste relates one person to various things. The verb
to dislike means the distaste relation.


Clark dislikes the beans. Lewis dislikes the bread.
```

Since we've defined this as a relation, we could change what the characters like and dislike during the course of the game, freely; for instance, characters might grow hungry and suddenly like all the edible articles.

```
Test me with "x lewis / x clark / clark, give the slippers to
lewis / clark, give the bread to lewis".
```

Example **The Man of Steel Excuses Himself** *RB*

Elaborating the report rules to be more interesting than "Clark goes west."

Report rules can be a good point at which to add local color: while Inform provides default descriptions of character behavior, these are rather generic and can stand to be customized.

For instance, if we wanted to liven up our previous Clark Kent example:

```
Report Clark Kent going a direction (called the way):
    say "[one of]With a particularly weak
excuse[or]Muttering[at random] about [random excuse
subject], Clark heads [way]." instead.


To say random excuse subject:
    choose a random row in the Table of Lame Excuses;
say "[reply entry]".


Table of Lame Excuses
reply
"needing a paper-clip"
"wanting an English-Tuvalu dictionary"
"walking a neighbor's dog"
"hearing air-raid sirens"
"having drunk too much coffee"
"thinking he smells smoke"
"wondering where Lois got to"
"needing to speak to Jimmy"
"noticing the Good Year blimp"
```

It's good to be careful, as the library report rules have been designed and tested to describe every contingency (going through doors, going in vehicles, etc.): so when replacing a report rule, we should try to consider all the possible variations of the action that we might want to describe.

However, in this case, our scenario is so simple that there are no doors, vehicles, or pushable objects, so we're safe in giving Clark a very simple reporting scheme.

ExampleFate Steps In *RB*

Fate entity which attempts to make things happen, by hook or by crook, including taking preliminary actions to set the player up a bit.

One of the nice things about before rules for actions is that they allow us to express some planning for characters other than the player: we've already seen how this works, a bit. But we could also use before rules to write plans for an abstract story-driving entity, rather than for other individual characters. This story-driver could be in charge of all the non-player characters, as well as spontaneous or natural changes in the environment, shaping the narrative around the player's behavior.

The following example is a very simple one, but the same concept could be worked out in a great deal more complexity, with all sorts of alternative procedures available to our story-manager:

"Fate Steps In"

Fate is a woman. After deciding the scope of the player: place Fate in scope. The description of Fate is "Not smiling." Instead of doing something other than examining to fate: say "As if."

Every turn: try fate tripping.

Tripping is an action applying to nothing.

Carry out someone tripping:
    if something dangerous (called the trap) is in the location:
        say "Lise chooses this moment to lick her fingers -- it's not gross, it's natural, you decide -- stand up, and head for the door. Unfortunately, her path crosses directly over [the trap]. There is a vaudevillesque moment where you try to warn or catch her; the next moment she's on the floor, looking shocked and also in quite a lot of pain. 'I'm not sure,' she says to you steadily but with unfocused gaze, 'but I think I might have broken my tailbone.'";
        end the story saying "Well, she's paying attention to you now".

Before someone tripping when the location does not contain a dangerous thing: try the person asked making a mess instead.

Making a mess is an action applying to nothing.

Carry out someone making a mess:
    let calamitous object be a random visible supporter which supports at least three things;
    if calamitous object is a supporter:
        say "[The calamitous object] tips over, spilling [the list of things on calamitous object] all over the place.";
        move the calamitous object to the location;
        now every thing on the calamitous object is in the location.

Definition: a thing is dangerous if it is not the carton and it is not the table and it is not a person.

Before someone making a mess when a safe supporter (called target) is visible:
    if Lise carries something, try Lise putting a random thing carried by Lise on the target instead.

Instead of someone making a mess when the tray is on the table:
    say "Just at that moment, a large blond man-thing in a red jacket walks more or less through you, and you come into violent contact with the table, knocking [the list of the things on the table] onto the floor.";
    now every thing on the table is in the location;
    now every thing on the tray is in the location instead.

Definition: a supporter is safe if the number of things on it is less than two.

McQuerry Dining Hall is a room.

The table is scenery in the dining hall. The table is a supporter.

Lise Fitzwallace is a woman in the Dining Hall. "Lise is at the nearest table, not apparently paying any attention to you." The description of Lise is "A capella singer, women's rugby champion, general object of attention from all genders. Unlikely to notice you unless fate smiles broadly." Lise carries a fork, a napkin, an empty glass, and a plate of half-eaten eggplant parmesan.

Report Lise putting something on something:
    say "Lise, still deep in thought, absently puts [the noun] on [the second noun]." instead.

The carrying capacity of the player is 2. The carton of chocolate milk is in the Hall. "There's a carton of milk beside you, which you set down for a moment -- but you do want it."

Instead of taking something when the player carries the tray:
    say "You've got both hands full with this tray."

The player carries the tray. On the tray is some macaroni and some overdone chicken. The macaroni and the chicken are edible. The tray is portable.

Test me with "get milk / put tray on table / get milk".

Test again with "drop tray".

210 **Example** **Spellbreaker** *RB*

P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

The answer is easy, but there is a trap:

```
Check saving the game when the location is the Vault:
say "That spell does not work here." instead.
```

The trap is that "Before saving the game...", which might have been our first guess, does not work: because out of world actions are exempt from Before, Instead and After rules.

"Spellbreaker" pulls this unpleasant, but in context witty, stunt as part of a situation which is engineered to force the player to reason through a weighing-objects puzzle using the perfect strategy rather than by guesswork. The illusion that the situation is fair - not rigged against the player, that is - would collapse if the player could save the game and keep retrying possibilities in the light of knowledge gained from earlier attempts. The moral of this story is that any attempt to use in-world situations to influence out-of-world commands should be *extremely uncommon*.

---

211 **Example** **A point for never saving the game** *RB*

In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

Here is one way to score this point with Inform:

```
Check saving the game for the first time: decrement the
score.
```

That has the right effect, but it just isn't sneaky enough. Instead let us quietly keep track of how many times the player saves:

```
Check saving the game: increment the number of
saves.
When play ends: if the number of saves is 0, increment
the score.
```

Sneakier, certainly, but now we could get the bonus even if the game ends earlier on, so finally:

```
When play ends: if the number of saves is 0 and the
score is 349, increment the score.
```

---

212 **ExampleCarnivale** *RB*

An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

Suppose we want to make an object that (unlike a backdrop) is definitely located in one room, but can be seen from far off. We want to allow the player to interact with it from a distance, but only using those actions that require visibility. Other actions should be denied:

```
"Carnivale"
```

```
The Fairground is a region. Park Entrance, By the
Wheel, and Candy Stand are in Fairground. Candy
Stand is north of By the Wheel. Park Entrance is west of
Candy Stand and northwest of By the Wheel.
```

```
The ferris wheel is scenery in By the Wheel. "It is
extravagantly tall and carries several dozen glass
gondolas for riders."
```

```
The description of By the Wheel is "You stand at the
foot of an enormous ferris wheel, which turns far too
quickly and never seems to stop for new riders."
```

```
The description of Park Entrance is "You are now just
inside the gates. Behind you snakes a triple line of
fairgoers all the way down the length of the valley to the
railway station. Roughly southeast of here is the ferris
wheel, towering over the other attractions."
```

```
The description of Candy Stand is "A hut in pale pink
and baby blue dispenses marshmallow death's-heads,
sugar-beetles, and other such treats. The giant ferris
wheel is just off to the south from here."
```

As the descriptions make clear, the ferris wheel should be visible from everywhere in the fair, so we'll borrow a line from the Activities chapter to make that happen:

```
After deciding the scope of the player:
    if the location is in Fairground, place the ferris wheel
in scope.
```

"Scope" determines what the player can interact with; by writing this rule, we make Inform understand all commands that refer to the ferris wheel when the player is anywhere in the fairground, instead of responding with

```
You can't see any such thing.
```

as it normally would.

Now, by default, if the player were to type TOUCH FERRIS WHEEL while in another room, he would get the response

```
You can't reach into By the Wheel.
```

This may not be quite what we want, but we can replace this text with our own reaching inside rule:

```
Rule for reaching inside a room:
    say "You can only look from this distance.";
    deny access.
```

And because our accessibility rules are considered before the "Instead" phase, we can write the following rule confident that it will apply only when the player is in fact in range to touch the ferris wheel:

```
Instead of touching the ferris wheel:
    say "You don't dare: it's spinning too fast."
```

```
Test me with "x ferris wheel / touch ferris wheel / se / x
ferris wheel / touch ferris wheel".
```

---

213 **ExampleEddystone** *RB*

Creating new commands involving the standard compass directions.

Using the compass directions in commands is a little bit finicky because directions are forbidden to figure in any interactions involving touch. (Really, directions are more a concept than an object; this is a compromise situation.) In any case, if we want to write a new command involving these, we need to be sure to specify that the direction is a visible thing. For instance:

```
"Eddystone"
```

```
The Lighthouse is a room. "A lonely place, but in these
tense times, no one but the lighthouse keeper and a few
trusted agents are allowed on the grounds at all, for fear
of sabotage."
```

```
The light is a fixed in place thing in the Lighthouse. "At
the center of the room is the light itself, a 1000-Watt
tungsten halogen light powered by diesel generator, and
having a visible range of twenty-six nautical miles."
Understand "lamp" as the light. It is lit. The light has a a
direction called heading. The heading of the light is
north.


A room is usually dark.


Understand "turn [something] [a direction]" as
reorienting it to. Reorienting it to is an action applying to
two things.
```

will give us

```
>turn light northeast
You must name something more substantial.
```

To avoid this mystifying result:

```
"Eddystone"


The Lighthouse is a room. "A lonely place, but in these
tense times, no one but the lighthouse keeper and a few
trusted agents are allowed on the grounds at all, for fear
of sabotage."


The light is a fixed in place thing in the Lighthouse. "At
the center of the room is the light itself, a 1000-Watt
tungsten halogen light powered by diesel generator, and
having a visible range of twenty-six nautical miles."
Understand "lamp" as the light. It is lit. The light has a
direction called heading. The heading of the light is
north.
```

```
A room is usually dark.


Understand "turn [something] [a direction]" as
reorienting it to. Reorienting it to is an action applying to
one thing and one visible thing.


Instead of turning the light, say "Try turning the light to
the direction of your choice."


Check reorienting it to: if the noun is not the light, say
"You couldn't do so meaningfully." instead; if the second
noun is up or the second noun is down, say "The light
only points in compass directions." instead.
```

And now that's done, we have a little fun calculating where the beam hits:

```
Carry out reorienting it to:
    now the heading of the light is the second noun;
    let way be the heading of the light;
    let place be the room way from the Lighthouse;
    while place is a room and place is lower than
Lighthouse:
        let place be the room way from the place;
    if place is not a room, now the beam is nowhere;
    otherwise move beam to the place.


Report reorienting it to: say "The light now points
[heading of the light][if the beam is in a room],
spotlighting [the holder of the beam][otherwise], into
empty space[end if]."


The beam is a lit thing. Understand "light" or "brilliant"
as the beam. "Brilliant light from the lighthouse floods
the whole area." It is fixed in place. Instead of doing
```

something other than examining to the beam: say "The light is, of course, intangible." The description is "The light is coming from the lighthouse, since the lamp is apparently pointed this way."

Altitude is a kind of value. 200 ft specifies an altitude. A room has an altitude. The altitude of a room is usually 50 ft. The altitude of the Lighthouse is 100 ft.

Definition: a room is low if its altitude is 20 ft or less.

The Jetty is south of the Lighthouse. "During daylight hours, a fine place to catch almost unlimited supplies of crayfish. Less entertaining by night."

North of the Lighthouse is the Uphill Road. The altitude of Uphill Road is 75 ft. North of Uphill Road is Hilltop. The altitude of Hilltop is 110 ft. The description of Hilltop is "The highest natural point around for miles; sometimes you will sit up here and watch for the lighthouse supply ship, the Lady Loch."

Northeast of the Lighthouse is Open Field. East of the Lighthouse is Stanley Creek Valley. The description of Stanley Creek is "This place used to have some other name meaning Ghost Valley in the aboriginal language, but it was piously renamed by missionaries." Train Trestle is east of the Stanley Creek Valley. "The now-abandoned track of the Bush Pacific Railway runs here, above Stanley Creek." The altitude of Train Trestle is 100 ft.

Before going from a room (called source) to a room (called destination):
    if source is lower than destination:
        say "It's an uphill climb...";

```
    otherwise:
        if destination is lower than source, say "You're
heading downhill now...";
        otherwise say "It's a straight shot."



Test me with "turn lamp / turn lamp down / turn lamp
east / e / e".
```

All very loosely based on the Eddystone Point lighthouse of Tasmania, built in 1889, and forbiddingly remote even today. George Isaacs, a child growing up in the lighthouse, remembers the plentiful crayfish.

---

214 Example Slogar's Revenge *RB*

Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

Under most circumstances, locking and unlocking require the player to be carrying the key he uses to unlock something. This makes sense -- unless the key is on a keychain, or on a chain around his neck, for instance. So here we explore one way to circumstantially override the carrying requirements, while still making sure that the player cannot unlock the door if the unlocking tool is nowhere in sight.

In essence, we are rewriting the carrying requirements rule with a different one of our own devising, and swapping it in only at those moments when it is correct to do so.

```
"Slogar's Revenge"



Section 1 - Procedure



The amulet carrying rule substitutes for the carrying
requirements rule when locking something with the
Amulet of Tumblers.
```

```
The amulet carrying rule substitutes for the carrying
requirements rule when unlocking something with the
Amulet of Tumblers.
```

We can now replace the usual behavior of the carrying requirements rule (to check whether the player is carrying something and, if not, to generate an implicit take) with a similar rule of our own; note that "if the player has the second noun" is a more compact way to write "if the player carries the second noun or the player wears the second noun":

```
This is the amulet carrying rule:
    if the player has the second noun:
        continue the action;
    say "(first picking up the amulet)[command
clarification break]";
    try silently taking the second noun;
    if the player is not carrying the second noun:
        stop the action;
```

```
Section 2 - Scenario
```

```
The Daunting Dungeon is a room.
```

```
West of the Daunting Dungeon is the Disturbing Door.
The Disturbing Door is a door. West of the Disturbing
Door is the Fallow Field.
```

```
The Disturbing Door is closed and locked.
```

```
The player wears the Amulet of Tumblers. The Amulet
of Tumblers unlocks the Disturbing Door.
```

```
Test me with "unlock disturbing door with amulet / open
door / west / remove amulet / close door / lock
```

```
disturbing door with amulet / drop amulet / unlock
disturbing door with amulet".
```

For a more systematic handling of the keychain problem (and a number of other refinements to the behavior of doors), see the Locksmith extension included with Inform.

Example **Magneto's Revenge** *RB*

Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

```
"Magneto's Revenge"

The School for Gifted Youngsters is a room. Kitty Pryde
is a woman in the School for Gifted Youngsters. "Kitty
Pryde waits for you to say why you summoned her out
of class." The description is "You see nothing special
about Kitty Pryde. But that is only because she has
learned to conceal her mutant powers: in fact she has
the alternative name Shadowcat, and the ability to
phase through solid objects."

The glass box is a container in the School for Gifted
Youngsters. It is closed, openable, and transparent. In
the box is a quantity of poisonous gas. In the box is a
thing called the message from Magneto.

Instead of examining the message when the player is
not carrying the message:
    say "You'd need to be holding it before you could
read it."

Instead of opening the glass box: say "You're deterred
by the swirling bottle-green mist within."
```

```
Check someone taking the gas (this is the gaseous
object rule): rule fails.


Unsuccessful attempt by someone taking the gas: say
"The gas isn't something one can pick up in one's bare
hands."


Every turn:
    if the player can touch the gas:
        say "The gas has reached your lungs!";
        end the story.


A rule for reaching inside something:
    if the person reaching is Kitty, allow access.


Persuasion rule for asking someone to try doing
something: persuasion succeeds.


Test me with "get message / kitty, get message / open
box / kitty, open box".
```

Now the player can get Kitty to take the message without releasing the poisonous gas:

```
Kitty Pride waits for you to tell her why you summoned
her out of class.


You can also see a glass box (closed) (in which are
some poisonous gas and a message from Magneto)
here.
```

```
>kitty, get message
Kitty Pryde picks up a message from Magneto.
```

ExampleWaterworld *RB*

A backdrop which the player can examine, but cannot interact with in any other way.

It's tempting to handle the player's inability to interact with something with a simple instead rule:

```
"Waterworld 1"

A view is a kind of backdrop. Instead of doing something
other than examining to a view, say "You are too far
from [the noun] to do anything but look."

The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."

The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.

Test me with "x sun / get sun / n / x sun / n / x sun".
```

Unfortunately, the rule does not address the case where the object in question is the second noun; so for instance the following example reveals the difficulty:

```
"Waterworld 2"
```

```
A view is a kind of backdrop. Instead of doing something
other than examining to a view, say "You are too far
from [the noun] to do anything but look."


The player carries a rope.


The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."


The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.


Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to the sun".
```

...where the response here behaves as though the sun is in reach. If we had a fully implemented tying action, the player would (even more disastrously) be allowed to lasso celestial objects.

We could add a second instead rule as well:

```
"Waterworld 3"


A view is a kind of backdrop.


Instead of doing something other than examining when
the noun is a view:
    say "You are too far from [the noun] to do anything
but look."
Instead of doing something other than examining when
the second noun is a view:
```

```
    say "You are too far from [the second noun] to do
anything but look."

The player carries a rope.

The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."

The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.

Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to sun".
```

This produces acceptable output again, but there is a more elegant way, one that works better with Inform's existing world model. Currently the default model assumes that accessibility -- whether the player can reach something or not -- is checked between the Before... rules and the Instead... rules. We can add our own accessibility rules, including this one to govern whether views are accessible. So for instance:

```
"Waterworld 4"

A view is a kind of backdrop.

The can't touch views rule is listed before the access
through barriers rule in the accessibility rulebook.

Accessibility rule (this is the can't touch views rule):
    if the action requires a touchable noun and the noun
```

```
is a view:
        say "You are too far from [the noun] to do anything
but look." instead;
    if the action requires a touchable second noun and
the second noun is a view:
        say "You are too far from [the second noun] to do
anything but look." instead;
```

```
The player carries a rope.
```

```
The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."
```

```
The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.
```

```
Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to sun".
```

Now our new accessibility rule fits into its proper stage.

ExampleDinner is Served *RB*

A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

```
"Dinner is Served"
```

```
Street in Kolonaki is a room. "There is a single round
table out on the street here, and a window more or less
at knee level looks down into the Olive Tree Gyro Shop,
```

```
which is partly basement."


The Street contains a round table. The table is scenery.
On the round table is a plate. On the plate are a gyro
and a mound of fresh potates. The plate is portable. The
potates and the gyro are edible. The description of
potates is "They'd be called french fries, at home, but
these are steak-cut and fried in olive oil." The
description of the gyro is "Dripping garlic-yogurt sauce."


Olive Tree Gyro Shop is inside from Street in Kolonaki.
Kostis is a man in the Gyro Shop. In the Shop is a
stand. On the stand is a rotating column of cooking lamb
flesh. In the shop is a closed, openable container called
a drinks refrigerator. The refrigerator contains a can of
Mythos beer and a can of Coke Light.
```

Here's the part that allows reaching through the window.

We replace the usual rule that says the player can never reach into a room with one that more specifically checks whether we are trying to reach through the window. If we aren't, we return the usual refusal. If we are, we return a custom refusal if the window is closed ("You can't reach through the closed window"), but allow access if the window is open.

```
The can't reach through closed window rule is listed
instead of the can't reach inside rooms rule in the
reaching inside rules.


This is the can't reach through closed window rule:
    let reaching through the window be false;
    if the container in question is a room and the
container in question is not the location:
        if the container in question is the Street and the
location is the Olive Tree Gyro Shop:
            now reaching through the window is true;
        if the container in question is the Gyro Shop and
the location is the Street:
            now reaching through the window is true;
        if reaching through the window is true:
            if the window is closed:
```

```
            say "You can't reach through the closed
window.";
            deny access;
        otherwise:
            allow access;
    otherwise:
        say "You can't reach into [the container in
question] from here.";
        deny access.
```

And the rest is window-dressing.

```
After looking when a room (called the next room) is
adjacent:
    try examining the next room.
```

```
Instead of examining a supporter, say "On [the noun]
[is-are a list of things on the noun]." Instead of
examining an open container, say "In [the noun] [is-are
a list of things in the noun]."
```

```
The window is a backdrop. It is in the Street and the
Shop. The window can be openable. The window can
be open. The window is openable and closed. Instead of
searching the window in the Street: try examining the
shop. Instead of searching the window in the Shop: try
examining the street.
```

```
Understand "examine [any adjacent room]" as
examining.
```

```
Instead of examining a room:
    say "Over in [the noun], you can see [a list of visible
things in the noun]."
```

```
After deciding the scope of the player:
    if the player is in the Street, place the Shop in scope;
```

```
    if the player is in the Shop, place the Street in scope.


Test me with "examine shop / open refrigerator / open
window / examine shop / open refrigerator / get beer / in
/ examine street / out / get gyro / close window / put
gyro in refrigerator / open window / put gyro in
refrigerator".
```

ExampleFlashlight *RB*

Visibility set so that looking under objects produces no result unless the player has a light source to shine there (regardless of the light level of the room).

```
"Flashlight"


The Schoolhouse is a room. "Though normally
comfortable, the room is dark and menacing during the
storm; rain sheets on the windows, and you can barely
see the flash of the lighthouse only a few miles away."


The cabinet is a fixed in place openable container in the
Schoolhouse. The hurricane lantern is a thing in the
Schoolhouse. "A hurricane lantern hangs from a peg."
The lantern is lit.


Visibility rule when looking under something:
    if the player is carrying a lit thing (called lamp):
        say "You shine [the lamp] under [the noun]...";
        there is sufficient light;
    there is insufficient light.


There is a marble. The marble can be found or lost. The
```

```
marble is lost.


Instead of looking under the cabinet when the marble is
lost:
    move the marble to the player;
    now the marble is found;
    say "Billy's lost marble! So that's where it got to!"


Test me with "look under cabinet / get lantern / look
under cabinet".
```

Because visibility is checked before instead rules, this discovery will (correctly) occur only when the player does have enough light.

---

219 ExampleBosch *RB*

Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports.

We could, if we wanted, make a table of stored actions all of which represent things that will earn points for the player. For instance:

```
"Bosch"


Use scoring.


The Garden of Excess is a room. The gilded lily is an
edible thing in the Garden of Excess.


The Pathway to Desire is west of the Garden of Excess.
The emerald leaf is in the Pathway.
```

```
Table of Valuable Actions

relevant action                point value
taking the emerald leaf        15
eating the gilded lily         5
```

(And our list would presumably continue from there, in the full game.)

```
The maximum score is 25.


After doing something:
    repeat through Table of Valuable Actions:
        if the current action is the relevant action entry and
turn stamp entry is less than 0:
            now the turn stamp entry is the turn count;
            increase the score by the point value entry;
    continue the action.


Understand "full score" or "full" as requesting the
complete score. Requesting the complete score is an
action out of world.


Check requesting the complete score:
    if the score is 0, say "You have not yet achieved
anything of note." instead.


Carry out requesting the complete score:
    say "So far you have received points for the following:
[line break]";
    sort the Table of Valuable Actions in turn stamp
order;
    repeat through the Table of Valuable Actions:
        if the turn stamp entry is greater than 0:
            say "[line break] [relevant action entry]: [point
value entry] points";
```

```
say line break.
```

```
Test me with "eat lily / w / full score / get leaf / full".
```

This system is tidy, but limited: we cannot give actions interesting names in the score list, like "seducing the pirate's daughter" or "collecting a valuable artifact". So it will not be ideal in all situations, but it has the virtue of being easy to extend, and of listing all of the player's successes in the order in which they occurred in his play-through.

---

220 Example Cactus Will Outlive Us All *RB*

For every character besides the player, there is an action that will cause that character to wither right up and die.

```
"Cactus Will Outlive Us All"

Death Valley is a room. Luckless Luke and Dead-Eye
Pete are men in the Valley. A cactus is in the Valley.
Persuasion rule: persuasion succeeds.

A person has an action called death knell. The death
knell of Luckless Luke is pulling the cactus. The death
knell of Dead-Eye Pete is Luke trying dropping the
cactus.

Before an actor doing something:
    repeat with the victim running through people in the
location:
        let the DK be the death knell of the victim;
        if the DK is not waiting and the current action is the
DK:
            say "It looks as if [the DK] was the death knell
for [the victim], who looks startled, then nonexistent.";
            now the victim is nowhere.
```

If we leave it at that, then pulling the cactus will kill Luckless Luke but then say "Nothing obvious happens.", which seems like a bit of an anti-climax. So we add a special case response for that one:

```
After pulling the cactus when Luckless Luke was in the
location:
    say "That's a real shame."

Test me with "get cactus / drop cactus / luke, get cactus
/ luke, drop cactus / pull cactus / look".
```

Example **Actor's Studio** *RB*

A video camera that records actions performed in its presence, and plays them back with time-stamps.

Here we construct a video camera to track and play back actions:

```
"The Actor's Studio"

Section 1 - The Video Camera

The video camera is a thing carried by the player.

Table of Videotape
time-stamp    recorded action
9:00 AM       waiting
with 25 blank rows.
```

Mode is a kind of value. The modes are idle, recording, and playing back. The video camera has a mode. The video camera is idle.

Understand "play back" as playing back. Instead of switching on the camera, try tuning the camera to recording. Instead of switching off the camera, try tuning the camera to idle.

The description of the video camera is "It is currently [mode]; its available settings are idle, recording, and playing back."

Understand "set [camera] to [a mode]" as tuning it to. Tuning it to is an action applying to one thing and one mode.

Instead of setting the camera to something:
    say "The available settings are idle, recording, and playing back."

Check tuning it to:
    if the noun is not the camera, say "Only the video camera can be set to [the mode understood]." instead.

Carry out tuning it to:
    now the mode of the noun is the mode understood.

Report tuning it to:
    say "You set [the noun] to [mode understood]."

After an actor doing something when the video camera

```
is recording:
    if the current action is tuning the video camera to
recording, make no decision;
    if the number of blank rows in the Table of Videotape
is greater than zero:
        choose a blank row in the Table of Videotape;
        now the recorded action entry is the current action;
        now the time stamp entry is the time of day;
    otherwise:
        now the video camera is idle;
        say "The video camera runs out of recording
memory and switches off.";
    continue the action.



Every turn when the video camera is playing back:
    say "On the camera screen, you see [run paragraph
on]";
    let starting playback be false;
    repeat through the Table of Videotape:
        if the recorded action entry is not waiting:
            now starting playback is true;
            say "[line break] -- [if the actor part of the
recorded action entry is the player]you [end if][the
recorded action entry], time stamped at [time stamp
entry]";
            blank out the whole row;
    if starting playback is false, say "only static.";
    otherwise say paragraph break.



Section 2 - The Scenario



The Actor's Studio is a room. Lucas is a man in the
Actor's Studio. Persuasion rule: persuasion succeeds.



The Studio contains an edible thing called a croissant.



Test me with "set camera to recording / x lucas / lucas,
take inventory / lucas, eat croissant / set camera to
playing back / z".
```

Notice that both Lucas' implied taking action (picking up the croissant) and his eating action are recorded on the same move.

Example **Anteaters** *RB*

The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing actions that apply to topics, as in "look up anteater colonies in the guide".

```
"Anteaters"

A book is a kind of thing. Understand "book" as a book.
A book has a table name called the contents.

Report consulting a book about:
    say "You flip through [the noun], but find no reference
to [the topic understood]." instead.

Instead of consulting a book about a topic listed in the
contents of the noun:
    say "[reply entry][paragraph break]".

The Guide to Desert Fauna is a book. The contents of
the Guide is the Table of Critters.

Table of Critters

topic                reply
"spines"             "You flip through the Guide for a while and eventually realise that
                     spines are flora, not fauna."
"the giant anteater" "The giant anteater, which grows to six feet in size and can kill a
                     jaguar, is a solitary animal, found in many habitats, including
```

grasslands, deciduous forests and rainforests. It does not form colonies. That's ants. They're actually quite easy to tell apart."

Death Valley is a room. The Guide is in the Valley.

The gizmo is in Death Valley. The gizmo has an action called idea. The description of the gizmo is "The gizmo is hard to describe, but it projects an idea of [idea]."

Before when the player carries the gizmo and the idea of the gizmo is waiting:
    say "[The gizmo] eagerly soaks up the whole idea of [the current action].";
    now the idea of the gizmo is the current action.

After dropping the gizmo:
    say "The percussion of the fall seems to have shaken the gizmo's idea loose! There's nothing for it now but [idea of the gizmo].";
    try the idea of the gizmo;
    now the idea of the gizmo is waiting.

Test me with "get guide / look up spines in guide / x gizmo / get gizmo / i / x gizmo / drop gizmo / get gizmo / look up anteater colonies in guide / x gizmo / drop gizmo".

Chapter 13: Relations

.8. The built-in verbs and their meanings; *§13.9. Defining new assertion verbs*; *§13.10. Defining new pr*

Contents of *Writing with Inform*
Chapter 12: Advanced Actions
Chapter 14: Adaptive Text and Responses
Indexes of the examples

§13.1. Sentence verbs

Descriptions of things - "open door", "people in the Drawing Room" - have already had a whole chapter to themselves. But descriptions are only half of the story of Inform's highly flexible language for talking about places, things and circumstances: this chapter is the other half, and is about the "sentence". Of course all text is made up of sentences, but Inform has a more specific meaning than that. Consider the following pieces of source text:

```
The mouse is in the teapot.
```

```
Every turn when the mouse is in the teapot, say "A tail hangs out of
the spout."
```

```
Instead of taking the mouse:
    say "The mouse slips from your hand and disappears into the
teapot!";
    now the mouse is in the teapot.
```

What these three extracts have in common is the sentence "the mouse is in the teapot". Such a sentence can be used in three different ways: to declare the original state of the world, to ask during play if the world currently has that state, or to change things during play so that it does.

Actually, though, only definite sentences about the present can be used in all three ways. A vague instruction like

```
now Mr Darcy can see the mouse;
```

will fail, because there are so many ways in which Darcy might be able to see the mouse that Inform has no way to know how to arrange matters. And this by contrast is not merely difficult but impossible:

```
now Mr Darcy has never seen the mouse;
```

Which cannot be arranged because the past cannot be changed.

Verbs also turn up inside the more complicated descriptions. For instance,

```
things which are in the teapot
people who can see the mouse
```

are both descriptions, not sentences, but they contain "to be" and "to be able to see" respectively.

This chapter is about the verbs which can be used in sentences and descriptions. Inform involves many other features which use verbs - the action "taking the mouse" and the phrase "end the story" both use forms of verbs (to take and to end) - but this chapter has nothing to do with them: so for the sake of clarity, we will call verbs that occur in sentences "sentence verbs".

Start of Chapter 13: Relations

Back to Chapter 12: Advanced Actions: §12.21. Guidelines on how to write rules about actions

Onward to §13.2. What sentences are made up from

§13.2. What sentences are made up from

A sentence consists of two nouns with a verb between them. Usually, the two nouns are descriptions, as in:

```
Mr Collins is in a lighted room.
```

Here "Mr Collins" and "a lighted room" are descriptions. But there are sentences where one or both of the nouns is a value of some other kind. For instance, in

```
if the score is greater than 10, ...
```

the sentence "the score is greater than 10" consists of two number values ("the score" and "10") connected by a verb part ("is greater than").

This chapter is about getting the most out of sentences by defining new verbs to express ideas not already built in to Inform. Before we can define a new sentence verb, however, we must first look at the meanings of verbs: which Inform calls "relations".

Start of Chapter 13: Relations

Back to §13.1. Sentence verbs

Onward to §13.3. What are relations?

Example 223: ★★★ **Formal syntax of sentences** A more formal description of the sentence grammar used by Inform for both assertions and conditions.

---

§13.3. What are relations?

Relations are what sentences express. They are yes/no questions about pairs of things: for example, to say that the coin is in the purse is to say that a particular relation ("being in") is true about a specific pair of things (the coin, the purse). It is neither a fact about the coin nor about the purse, but about the two together.

Inform comes with a number of relations built in, almost all of which have been used in previous chapters already. The following table names some of the more useful ones, giving examples of sentences to bring them about:

```
containment relation - The coin is in the purse.
support relation - The coin is on the table.
incorporation relation - The coin is part of the sculpture.
```

carrying relation - The coin is carried by Peter.
wearing relation - The jacket is worn by Peter.
possession relation - if Mr Darcy has a rapier...
adjacency relation - The Study is east of the Hallway.
visibility relation - if Darcy can see Elizabeth...
touchability relation - if Darcy can touch Elizabeth...

These relation names do not trip off the tongue, but they relatively seldom need to be referred to.

The same meaning can often be expressed by using several different verbs, or using the same verb in several different ways, as in the following examples:

The coin is in the purse.
The purse contains the coin.
The coin is contained by the purse.

all of which boil down to saying that the coin and purse satisfy the containment relation. Because of that, *relations are not the same as verbs*. To create a new idea, we will need first to create a new relation, and only then can we set up a verb which allows us to talk about that relation.

Start of Chapter 13: Relations
Back to §13.2. What sentences are made up from
Onward to §13.4. To carry, to wear, to have

§13.4. To carry, to wear, to have

Inform has altogether five mutually exclusive ways in which one thing can be physically joined to another one:

containment relation - The coin is in the purse.
support relation - The coin is on the table.
incorporation relation - The coin is part of the sculpture.
carrying relation - The coin is carried by Peter.
wearing relation - The jacket is worn by Peter.

This is why we cannot have

```
The coin is on the table.
The coin is part of the table.
```

simultaneously, and it is a rare exception to the general rule that having one relation does not affect having another.

But there is also a sixth relation used in Inform for these meanings: the possession relation, which is the meaning of the verb "to have". At first sight this looks the same as the carrying relation, but in fact it is a convenient shorthand for "carrying or wearing", provided for conditions rather than assertions:

```
if Mr Darcy has a wet shirt ...
```

will be true during play if he is either carrying or wearing the shirt.

Still another relation exists which can be tested, but not declared to be true or false: the concealment relation, which is the meaning of the verb "to conceal". So we can ask:

```
if Mr Darcy conceals a fob watch ...
```

Start of Chapter 13: Relations

Back to §13.3. What are relations?

Onward to §13.5. Making new relations

Example 224: **Celadon**
Using the enclosure relation to let the player drop things which he only indirectly carries.

Example 225: **Interrogation**
A wand which, when waved, reveals the concealed items carried by people the player can see.

---

§13.5. Making new relations

We can create new relations like so:

```
Loving relates various people to one person.
```

Every relation has a name which ends with the word "relation", and in this case the name is "loving relation". While the name is often just two words long, as here, it doesn't have to be:

```
Adept sensitivity relates one person to one vehicle.
```

makes the "adept sensitivity relation". (The limit is 32 words.)

In such a definition, we have to say what kind of thing appears on the left and right of any relation, and also whether "one" or "various" possibilities can exist. In the example

```
Loving relates various people to one person.
```

what we are saying is that only people love; that they only love people; and that each person loves only one other person (at any given moment).

The "various" part comes in because, for instance, we might have:

```
Verenka loving relation Stankevich
Liubov loving relation Stankevich
```

so that various people (Verenka and Liubov, to name but two) love one person (Stankevich). But we are forbidding anyone to love two other people at the same time: Stankevich must decide which of them to love, or pick someone else, or no-one at all. Similarly, we would not allow

```
Liubov loving relation Belinsky
```

It is sometimes convenient to give a name to the other side of a relationship, so to speak. We might imagine:

```
Pet-ownership relates various animals to one person (called the
owner).
```

It would then make sense to talk about "the owner of Loulou", and we could have phrases like "now Flaubert is the owner of Loulou" or "if the owner of Loulou is a woman..." and so forth. This, however, would not be allowed:

```
Pet-ownership relates various animals (called the pet) to one person.
```

because "the pet of Flaubert" would be ambiguous: he might have owned dozens.

Start of Chapter 13: Relations
Back to §13.4. To carry, to wear, to have
Onward to §13.6. Making reciprocal relations

§13.6. Making reciprocal relations

The relationships described in this chapter so far are by no means always reciprocated. For instance, if a stone is on a table, then it is never true that the table is also on the stone. And the question may not even be meaningful to ask. If Peter wears a jacket, the jacket does not even have the possibility of wearing Peter.

But sometimes we do want a relation which works both ways equally well. These are simple to set up:

```
Meeting relates people to each other.
```

The effect is that various people know various other people, and this is always reciprocated. If Daisy knows Sophie then, automatically, Sophie knows Daisy. This even-handedness is maintained throughout play, so that whatever changes are made it is always true that if A knows B then B knows A.

And similarly for a reciprocal relation between one and another:

```
Marriage relates one person to another.
```

In this case, we can again give a name to the partner under a relation:

```
Marriage relates one person to another (called the spouse).
```

and now, for instance, we may have that the spouse of John is Yoko and the spouse of Yoko is John.

Since many of these examples have involved people, it might be worth mentioning again that any kind can be involved, not just the "person" kind.

Start of Chapter 13: Relations

Back to §13.5. Making new relations

Onward to §13.7. Relations in groups

Example 226: ★★★ **Four Cheeses** A system of telephones on which the player can call distant persons and have conversations.

---

§13.7. Relations in groups

Finally, there is a kind of relation which binds even more strongly.

```
Nationality relates people to each other in groups.
```

This is a kind of relation which divides people up: we might wish to have all the Icelandic people related to each other, all the Peruvians to each other, and so on. If there were a Pacific island called Informia with one inhabitant, then that person would be related only to himself. As time goes by, we could imagine people emigrating, and so on, so that these groupings would switch: perhaps everyone would leave Belgium and, for a while, there would be no Belgian nationals at all.

The testing command RELATIONS prints out the current state of all the relations created in the source code. For instance:

```
>relations
Overlooking relates various rooms to various rooms:
   The Pub >=> the Garden
   The Garden >=> the Shrubbery
   The Shrubbery >=> the Sundial Plot
Friendship relates people to each other:
   Mr Wickham <=> Sophie
   Charlotte <=> Sophie
Marriage relates one person to another:
   Mr Wickham == Sophie
```

That can produce a lot of output. To see only a single relation, or to see it at some intermediate point in a calculation, there's also a testing phrase:

show relation (relation of values to values)

This phrase is for testing purposes only. It shows the current state of the named relation, that is, it shows which values relate to which other ones, where it's possible to do this in any sensible way.

But this is a phrase - not a typed command.

Start of Chapter 13: Relations

Back to §13.6. Making reciprocal relations

Onward to §13.8. The built-in verbs and their meanings

Example 227: ★ **Transmutations** A machine that turns objects into other, similar objects.

Example 228: ★★★ **Otranto** A kind of rope which can be tied to objects and used to anchor the player or drag items from room to room.

§13.8. The built-in verbs and their meanings

It is all very well to define new relations, but this does nothing if there is no way to assert that they are true, or to ask whether they are true or false. That requires a verb: in fact, a relation is nothing more than what Inform uses as the "meaning" of a verb. The assertion verbs built in to Inform have the following built-in relations as their meanings:

Verb - Relation
to be - equality relation
to have - possession relation
to contain - containment relation
to support - support relation
to carry - carrying relation
to wear - wearing relation
to incorporate - incorporation relation

Two of Inform's built-in relations are expressed using prepositions instead:

Preposition - Relation
to be part of - (reversed) incorporation relation
to be adjacent to - adjacency relation

It would be easy to make verbs for these if we wanted ("to adjoin", say) using the techniques of the next section.

The verb *to be* is grammatically different from any other, and its meaning is too complicated to be fully expressed by any one relation. A great deal of the Inform program is given over to its "meaning", which we are not allowed to change or imitate. The "equality relation" is simple enough, and is the one implied by conditions like

```
if the score is 20, ...
```

but *to be* can have more complicated implications - "if Mr Wickham is hungry" clearly doesn't test whether two quantities are equal. Fortunately the other verbs are much simpler.

There are a few other built-in verbs, as can be seen in the Index, but these are mostly for experts only. For example:

Verb - Relation
to mean - meaning relation
to provide - provision relation

"To mean" can be used to make new verbs, as we'll soon see. Provision is to do with whether something can have a given property: for example, "if R provides the property lighted" tests whether R is able to have this property, not whether it actually has it at the moment.

Start of Chapter 13: Relations

Back to §13.7. Relations in groups

Onward to §13.9. Defining new assertion verbs

§13.9. Defining new assertion verbs

Here is an example definition of a new verb:

```
The verb to sport means the wearing relation.
```

Once this is done, we can write the assertion

```
Mr Wickham sports a Tory rosette.
```

which will do the the same thing as

```
Mr Wickham wears a Tory rosette.
```

because both verbs have the same relation as their meaning.

Earlier versions of Inform needed to be told how to make other parts of the verb, but that's rarely true now. Just writing:

```
The verb to sport means the wearing relation.
```

is enough for Inform to understand "he sports", "they sport", "he sported", "it is sported", "he is sporting", "he had sported" and so on. It works with irregular verbs, too; it has a very comprehensive dictionary. But it's legal to spell out the conjugation if need be:

```
The verb to sport (he sports, they sport, he sported, it is sported)
implies the knowledge relation.
```

Occasionally it's convenient to have the relation the other way around. For instance:

```
The verb to grace means the reversed wearing relation.
```

With that defined, these two sentences have identical meanings:

```
Mr Wickham sports a Tory rosette.
A Tory rosette graces Mr Wickham.
```

Reversed in this sense means that the things related - the subject and object of the verb - are the other way round.

The Phrasebook index contains all the verbs associated with assertions, in the Verbs section. When we add new verbs to our source, those will appear in the Phrasebook as well.

The verbs above ("to grace", "to sport") are short ones, but we're free to make them longer than that. For example:

```
The verb to cover oneself with means the wearing relation.
```

```
Peter is covering himself with a tent-like raincoat.
```

Here we have "to cover oneself with", four words long; the limit is 29.

Start of Chapter 13: Relations

Back to §13.8. The built-in verbs and their meanings

Onward to §13.10. Defining new prepositions

Example 229: **Unthinkable Alliances** People are to be

grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give notice of quitting, and to become a lone wolf.

Example 230: ★★★ **The Unexamined Life** An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a nudge he wants at any given moment.

---

§13.10. Defining new prepositions

The term preposition is used here, a little loosely, to mean anything which we add to the verb *to be* in order to talk about some relation or other. We have seen many examples already, such as:

To be in - The ball is in the box.
To be part of - The lever is part of the slot machine.

These are defined just the way verbs are. Compare the following:

Suspicion relates various people to one person.

The verb to suspect means the suspicion relation.

The verb to be suspicious of means the suspicion relation.

The result of this is that

```
Hercule Poirot suspects Colonel Hotchkiss.
Hercule Poirot is suspicious of Colonel Hotchkiss.
```

are exactly equivalent, and so are these two descriptions:

```
somebody who suspects Colonel Hotchkiss
somebody suspicious of Colonel Hotchkiss
```

While most prepositions are short ("in", "part of", "suspicious of"), they're free to be longer if need be ("inordinately far away from"): the limit is 30 words, which should be ample.

We can also define verbs as auxiliaries, like so:

```
The verb to be able to approach means the approachability relation.
```

Now we can ask if Poirot "can approach" Hotchkiss, and so on.

Start of Chapter 13: Relations

Back to §13.9. Defining new assertion verbs

Onward to §13.11. Indirect relations

Example 231: ★ **The Abolition of Love** A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

Example 232: ★ **Swerve left? Swerve right? Or think about it and die?** Building a marble chute track in which a dropped marble will automatically roll downhill.

Example 233: ★ **Beneath the**

Surface An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

Example 234: ★★★ **Bogart** Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

---

§13.11. Indirect relations

We have already seen, in the chapter on Descriptions which is a forerunner of this one, that Inform provides not only "adjacent" as a way of seeing if one room is directly connected to another, but also "the best route from A to B", which allows us to see if any sequence of moves connects them.

Something similar - in fact, simpler - is allowed for any relation between objects. Suppose we would like to go sledging: we can go downhill, but not up. Some quite distant places may be reachable, while others close by may not be, even if lower than us, because they would involve climbing again at some point. The following would implement this:

```
Overlooking relates various rooms to various rooms.


The verb to overlook means the overlooking relation.


The Garden overlooks the Shrubbery. The Folly overlooks the
Garden. The Shrubbery overlooks the Sundial Plot. The Old Ice
House overlooks the Garden.


After looking:
    say "This wintry vantage point overlooks [the list of rooms
overlooked by the location].";
    let the way be the next step via the overlooking relation from the
```

```
location to the Sundial Plot;
    if the way is a room, say "To sledge downhill to the Sundial, aim for
[the way].";
    otherwise say "It is not possible to sledge downhill to the Sundial."
```

Here we're making use of:

next step via (relation of values to values) **from** (object) **to** (object) **... object**

This phrase tries to find a shortest route between the two given endpoints, using the given relation of objects to determine single steps. Example:

```
next step via the overlooking relation from the Folly to the
Chinese Lake
```

The result is the special object value "nothing" if the two endpoints are the same or if no route exists.

number of steps via (relation of values to values) **from** (object) **to** (object) **... number**

This phrase tries to find the length of a shortest route between the two given endpoints, using the given relation of objects to determine single steps. Example:

```
number of steps via the overlooking relation from the Folly
to the Chinese Lake
```

The result is 0 if the two endpoints are the same, or -1 if no route exists.

Another example would be the "six degrees of separation" game, where it is claimed that any two people on Earth are connected by a sequence of up to six acquaintances. In an Inform implementation, we might talk about "the next step via the friendship relation from George Bush to Saddam Hussein", for instance, a phrase likely to evaluate to Donald Rumsfeld, and then

```
the number of steps via the friendship relation from George Bush to
Saddam Hussein
```

would be... but that would be telling.

As with route-finding through the map, finding "the next step via" a relation can be slow. For instance, suppose we have dozens of articles of clothing all partially revealing each other, connected by two relations - overlying and underlying. Then "the next step via" these relations allows us to establish what can be worn on top of what else. If we need to calculate this often, and there are enormous wardrobes of clothes to choose from, speed starts to matter.

Once again there is a choice of algorithms: "fast" and "slow", where "fast" needs much more memory. To make route-finding for a given relation "fast", we have to declare it that way:

> Overlying relates various garments to various garments with fast route-finding.
> Overlapping relates various garments to each other with fast route-finding.

Otherwise, the "slow" method will be used.

This "with fast route-finding" note can only be added to various-to-various relations. (Although route-finding through various-to-one and one-to-various relations is fully supported, it exploits the relative simplicity of these problems to use a more efficient algorithm than either "fast" or "slow".)

See **Adjacent rooms and routes through the map** *for route-finding through the map rather than a relation*

Start of Chapter 13: Relations

Back to §13.10. Defining new prepositions

Onward to §13.12. Relations which express conditions

Example 235: **The Problem of Edith** A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before.

---

§13.12. Relations which express conditions

One last way to create a new relation and, in many ways, the easiest of all. If we write:

```
Contact relates a thing (called X) to a thing (called Y) when X is part
of Y or Y is part of X. The verb to be joined to means the contact
relation.
```

then we would be able to talk about a handle being joined to a door, and a door being joined to a handle, and so on. We are not allowed to declare:

```
The hook is joined to the line.
```

because the question of whether they are joined is not for us to decide: that will be for the condition to determine, whenever we test it. Similarly, we cannot meaningfully write

```
now the hook is joined to the line;
```

(and Inform will not let us) because this relation is not something we can force either way: we can make it come true by other means, maybe, but we cannot simply make it true by saying so. Lastly, this kind of relation is restricted in that we are not allowed to find paths or calculate numbers of steps through it.

So this way to define relations is, on the face of it, just a sort of verbal trick to write conditions in a more attractive way. The more flexible, changeable relations in previous sections have much greater expressive power. All the same, it is nice to be able to write -

```
Nearness relates a room (called A) to a room (called B) when the
number of moves from B to A is less than 3. The verb to be near
means the nearness relation.
```

and then to be able to write rules like:

```
Instead of listening when the location is near the Sundial: say "You
hear a splashing of water."
```

As with other relations, there's no reason why we have to use objects. For example:

```
Material is a kind of value. The materials are wood and metal. A thing
has a material.

Materiality relates a thing (called X) to a material (called Y) when Y is
the material of X. The verb to be made of means the materiality
relation.
```

which enables us to write:

```
if the cube is made of wood, ...
say "The carpenter looks at [the list of things which are made of
wood].";
```

And here is a mathematical one:

```
Divisibility relates a number (called N) to a number (called M) when
the remainder after dividing M by N is 0. The verb to divide means
the divisibility relation. The verb to be a factor of means the divisibility
relation.
```

We now find that "2 divides 12", "5 is not a factor of 12" and "12 is divisible by 3" are all true. Again, we are only really gaining a nice form of words, but improving the clarity of the source text is never a bad thing.

Start of Chapter 13: Relations

Back to §13.11. Indirect relations

Onward to §13.13. Relations involving values

Example 236: **Wainwright Acts** A technical note about checking the location of door objects when characters other than the player are interacting with them.

Example 237: **A Humble Wayside Flower** Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

---

§13.13. Relations involving values

Although most of the examples in this chapter have involved objects, relations can connect almost any values together. We can create relations in groups, one to various relations, various to one relations, one to one relations, and various to various relations for any combination of kinds. For example:

```
Partnership relates various texts to various texts.


The verb to belong with means the
partnership relation.


"cheese" belongs with "crackers".
"clam" belongs with "chowder".
```

How might we make use of this? Clearly it would be impractical to keep trying:

```
if "caviar" belongs with "aardvarks", ...
if "caviar" belongs with "abacuses", ...
...
```

to find out what "caviar" belongs with. It's still harder to find out if it belongs with anything at all -- in theory we would have to try every possibility, which of course is impossible. Instead we have these phrases:

if (value) **relates to** (name of kind) **by** (relation of values to values)**:**

This condition is true if the value V is such that V relates to something by the given relation. Example: suppose partnership relates various texts to various texts. Then we can test

```
if "chalk" relates to a text by the partnership relation, ...
```

if (name of kind) **relates to** (value) **by** (relation of values to values)**:**

This condition is true if the value V is such that something relates to V by the given relation. Example: suppose partnership relates various texts to various texts. Then we can test

```
if a text relates to "cheese" by the partnership relation, ...
```

If a partner does exist, then we can find it with:

(name of kind) **to which**/whom (value) **relates by** (relation of values to values) **... value**

or:
(name of kind) **that**/which/whom (value) **relates to by** (relation of values to values) **... value**

This phrase produces an Y such that the given value V relates to Y by the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

```
the text to which "chalk" relates by the partnership relation
```

which might be, say, "cheese". It's a run-time problem to use this if no such Y exists.

(name of kind) **that**/which/who **relates to** (value) **by** (relation of values to values) **... value**

This phrase produces an X such that X relates to the given value V by the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

```
the text which relates to "cheese" by the partnership relation
```

which might be, say, "chalk". It's a run-time problem to use this if no such X exists.

Of course, there might be many answers to this question, so perhaps these are neater:

list of (name of kind) **that**/which/who **relate to** (value) **by** (relation of values to values) **... value**

This phrase produces a list of all the X such that X relates to the given value V by the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

```
list of texts which relate to "cheese" by the partnership
relation
```

which might be, say, { "chalk", "grapes", "macaroni" }. The answer might be the empty set, but that's not a problem.

list of (name of kind) **to which**/whom (value) **relates by** (relation of values to values) **... value**

or:
list of (name of kind) **that**/which/whom (value) **relates to by** (relation of values to values) **... value**

This phrase produces a list of all Y such that the given value V relates to Y by the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

list of texts to which "chalk" relates by the partnership relation

which might be, say, { "cheese", "blackboard", "cliffs" }. The answer might be the empty set, but that's not a problem.

Finally, it's sometimes useful to get at the list of all values which can appear on the left or right hand side of a relation. We need tongue-twister like wording to do it, but:

list of (name of kind) **that**/which/whom (relation of values to values) **relates ... value**

This phrase produces a list of all X which relate to anything under the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

list of texts which the partnership relation relates

list of (name of kind) **to which**/whom (relation of values to values) **relates ... value**

or:
list of (name of kind) **that**/which/whom (relation of values to values) **relates to ... value**

This phrase produces a list of all Y which anything relates to under the given relation. Example: suppose partnership relates various texts to various texts. Then we can obtain

list of texts which the partnership relation relates to

For efficiency reasons, there are no guarantees about what order these lists have - but they can of course always be sorted when found.

Back to §13.12. Relations which express conditions
Onward to §13.14. Relations as values in their own right

Example 238: ★ **Meet Market** A case in which relations give characters multiple values of the same kind.

Example 239: ★★★ **For Demonstration Purposes** A character who learns new actions by watching the player performing them.

---

§13.14. Relations as values in their own right

As we've seen, most relations have names - "containment relation", for instance. These are themselves values in Inform, though there are a few restrictions on how they are used. (Relations can contain a colossal amount of data, so we don't want to have to copy them casually.)

Consider these two examples:

```
Parity relates a number (called N) to a number (called M) when N
minus M is even.
```

```
Joint magnitude relates a number (called N) to a number (called M)
when N plus M is greater than 7.
```

Here "parity relation" and "joint magnitude relation" are both values of the same kind: "relation of numbers to numbers". In general, every relation is a value of kind "relation of K to L", for the appropriate kinds K and L. So the parity relation doesn't have the same kind as the containment relation, for example. Because it often happens that K and L are the same, we can just say "relation of K" in this case, so we could equally say that the kind of the parity relation is "relation of numbers".

This is useful to know when writing phrases like so:

```
To chart (R - a relation of numbers):
    repeat with N running from 1 to 5:
        repeat with M running from 1 to 5:
            if R relates N to M, say "[N] <=> [M] ";
        say "[line break]";
```

and now "chart parity relation" will work nicely, but "chart visibility relation" will be rejected (as it should be, because it relates things, not numbers). In general, if R is any relation, we can write

```
if R relates X to Y, ...
now R relates X to Y;
now R does not relate X to Y;
```

to test, set and unset a relation R between two values. (Inform checks that the values X and Y have the right kind and produces a problem message if not.)

Several useful adjectives can be applied to relations:

```
"empty" - nothing relates to anything else
"symmetric" - by definition X relates to Y if and only if Y relates to X
"equivalence" - this is a relation "in groups", or an "equivalence
relation"
"one-to-one" - it relates one K to one L
"one-to-various" - similarly
"various-to-one" - similarly
"various-to-various" - similarly
```

So for example it's possible to ask

```
if R is a symmetric one-to-one relation of texts, ...
```

With some relations, it's possible to clear them out by writing:

```
now R is empty;
```

and with temporary relations (see the next section), it's even possible to change their valencies (one-to-one vs. one-to-various, etc.) using "now", but only when they are empty. The exceptions where "empty" can't be used are those which can't be changed at all, like the parity relation above, and a few built-in cases such as the support, containment and incorporation relations, where emptying would dissolve the

model world in a disastrous way.

Start of Chapter 13: Relations

Back to §13.13. Relations involving values

Onward to §13.15. Temporary relations

Example 240: ★ **Number Study** The parity and joint magnitude relations explored.

§13.15. Temporary relations

So far in this chapter, we've only seen relations which exist permanently during play. The relationships might change - sometimes Red Riding Hood would be in the Woodcutter's Cottage, sometimes not - but the relations themselves were eternal.

In fact, though, we can also create relations to be dynamic data structures, like lists:

let (a name not so far used) **be** (description of relations of values to values)

This phrase creates a new temporary variable, and sets its value to the identity of a newly created and equally temporary relation. These last only for the present block of phrases, which certainly means that they exist only in the current rule. Example:

```
let the password dictionary be a relation of texts;
```

This makes a purely temporary various-to-various relation between texts, which lasts as long as the temporary value "password dictionary" lasts. By default, relations are various-to-various, but we could instead write, say:

```
let the nicknames catalogue be a various-to-one relation of
texts;
```

Such a relation exists only in the current phrase, and is destroyed when the phrase finishes, like any other "let". Of course there's no verb whose meaning is this relation,

but that's no obstacle, because we can manipulate it using "relates":

```
now the nicknames catalogue relates "Trudy" to "Snake-eyes";
```

(At present such a relation cannot be used outside its own phrase.)

Start of Chapter 13: Relations

Back to §13.14. Relations as values in their own right

Onward to §13.16. What are relations for?

§13.16. What are relations for?

It is easy to say what verbs are for: they are to express relations. But what are relations for?

Inform 7's focus on relations between objects is unusual as an approach to interactive fiction; the concept does not exist in most design systems, or rather, it does but is submerged. Traditional design systems do, after all, have the spatial relations of being inside, on top of, and so on. It could well be said that these are the only relationships that inanimate objects ever have. A stone can be on top of a table, and if so then that expresses their entire association.

This is because the stone, and the table, have no opinions, emotions, knowledge or memory. If the stone is taken away and then put back, nothing has changed. People, on the other hand, tend to remember having met each other before; they like being in some places, but not others; their behaviour depends on who, or what, is nearby. Being conscious, they have internal states, unlike the stone. Relations are a simple but powerful way to express and talk about such connections, and although they have numerous uses in physical contexts too, they are at their most powerful when helping to make the characters of interactive fiction come alive.

Start of Chapter 13: Relations

Back to §13.15. Temporary relations

Onward to Chapter 14: Adaptive Text and Responses: §14.1. Tense and narrative viewpoint

Example 241: ★★ **Murder on**

the Orient Express A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but there is physical evidence which may change their minds...

Example 242: ★★ **What Not To Wear** A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

Example 243: ★★★ **Mathematical view of relations** Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

Example 244: ★★★ **Graph-theory view of relations** Some notes on relations from the point of view of graph theory.

Examples from Chapter 13: Relations

Start of this chapter

Chapter 14: Adaptive Text and Responses

Indexes of the examples

★★★ Example **Formal syntax of sentences**

A more formal description of the sentence grammar used by Inform for both assertions and conditions.

An entire grammar for the whole mass of Inform would not be linguistically interesting: it contains many convenient wordings which are not really part of a grand pattern. Inform does, however, have a formal notion of a Sentence, a grammatical structure which we shall call S. It is almost true that conditions ("if the flowerpot is on the wall") have the same grammar as assertions ("The flowerpot is on the wall") and "now" phrases ("now the flowerpot is on the wall"). All three use the S grammar, so we could define an assertion as "S.", say that "if S", "while S", "when S" and so on are conditions, and say that "now S" defines the "now" declaration.

Grammatical sentences do not necessarily make sense, of course. Many perfectly grammatical assertions in fact give rise to problem messages:

> The wicker basket is not in the kitchen. (*Unhelpfully negative.*)
> The wicker basket has been in the kitchen. (*Talks about a time which never existed.*)
> The wicker basket is full. (*Full of what? Too vague.*)
> The wicker basket is the ginger cat. (*Demonstrably false.*)

Whereas the first three, at least, would be sensible as conditions. So saying that assertions are "just like" conditions is a little misleading: what they have in common is S, the underlying grammar they each use as a starting-point.

To define S, we break it up into subsidiary structures. The most important is the Description Phrase (DP), examples of which include "the red basket", "somewhere lighted" and "an empty open container". Clearly sentences include DPs, but they also include other ingredients. The general pattern used in Inform is very simple:

> 1. S = DP + VP
> 2. VP = Verb + DP

where VP is another structure, the Verb Phrase. For instance:

> S (The horseman wears a riding helmet)
> = DP (The horseman) + VP (wears a riding helmet)
> VP (wears a riding helmet)
> = Verb (wears) + DP (a riding helmet)

In that example, the Verb was the single word "wears". More generally, Inform allows a Verb to include adverbs and prepositions, to be negated, and to come in any of four tenses, so the following are all valid examples of Verb in our grammar:

wore
carries
is carried by
had not been inside

Although we are not going through the definition of Description Phrases in detail, it is worth noticing how "which" and "who" behave:

3a. DP = DP + which + VP
3b. DP = DP + who + VP

Thus "an open container which is in the Ballroom" can be broken down as:

DP (an open container) + which + VP (is in the Ballroom)

To understand compounds like "something in a container", we have to invent a new grammatical structure for "in a container" and similar: let's call this a Relative Phrase (RP).

4. DP = DP + RP

Thus "an open container in the Ballroom" is DP (an open container) + RP (in the Ballroom). Relative Phrases have two different forms:

5a. RP = Preposition + DP
5b. RP = Participle + DP

so that "in a container" is an example of 5a. An example of 5b would be

RP (worn by Mr Darcy) = Participle (worn by) + DP (Mr Darcy)

That is nearly it, but not quite: we must go back to the "almost" in the statement above that assertions and conditions "almost" have the same grammar S. The difference arises from a curious irregularity in English called subject-verb inversion (see the *Oxford English Grammar* at 3.22F), whereby assertions can be reversed but not conditions. For instance,

In the Garden is a sunflower.

This does not follow the pattern S = DP + VP, because "in the garden" is not a DP: indeed, it is not a noun at all. To make sense of this sentence, Inform reverses it to "A sunflower is in the Garden", which does indeed follow DP + VP. Hence the final rule:

6 (assertions only). S = RP + Verb + DP

So the condition "if in the garden is a sunflower..." fails because rule 6 does not apply to the grammar for conditions: while occasional poetic uses of subject-verb inversion do turn up in conditions ("If On A Winter's Night A Traveller", say), they are rare in ordinary English usage, and illegal in Inform. That completes the S grammar, so to recap:

1. S = DP + VP
2. VP = Verb + DP
3a. DP = DP + which + VP
3b. DP = DP + who + VP
4. DP = DP + RP
5a. RP = Preposition + DP
5b. RP = Participle + DP
6 (assertions only). S = RP + Verb + DP

ExampleCeladon

Using the enclosure relation to let the player drop things which he only indirectly carries. *RB*

By default, Inform only lets the player drop those things which he is carrying -- that is, those directly in his possession. Things inside satchels

or on portable trays have to be taken first.

If we want to change this behavior, we might add a dropping rule that distinguishes between carrying and mere enclosure (introduced back in "The location of something" in the chapter on Things):

```
"Celadon"

The Tea Room is a room. The player carries a black
lacquer tray. The lacquer tray is portable. On the
lacquer tray are a celadon teapot and a napkin.

Before dropping something:
    if the player does not carry the noun and the player
encloses the noun:
        say "(first taking [the noun] from [the holder of the
noun])[command clarification break]";
        silently try taking the noun;
        if the player does not carry the noun, stop the
action.

Instead of taking the napkin:
    say "It seems to be stuck to the tray, possibly by an
underlying wad of gum."

Test me with "i / drop teapot / i / look / drop teapot / drop
napkin / i / drop tray".
```

ExampleInterrogation *RB*

A wand which, when waved, reveals the concealed items carried by people the player can see.

"Interrogation"

The X-Ray Vision Wand is carried by the player.

Instead of waving the X-Ray Vision Wand:
    say "Disappointingly, nothing happens."

Instead of waving the X-Ray Vision Wand when the player can see someone who is concealing something:
    say "The wand glows green. Immediately you see on the monitor [a list of things which are concealed by people who can be seen by the player]."

After printing the name of a thing (called target) which is carried by someone while waving the wand:
    say " (carried by [a random person who carries the target])"

The Interrogation Chamber is a room. "Despite its gothic name, this room is a rather civilized place for your work, with large plate-glass windows and a fitted carpet."

A thing can be secret or obvious. A thing is usually obvious.

Brian is in the Interrogation Chamber."Brian lounges against the wall." Brian carries a quantity of plastic explosive. The explosive is secret.

Janine is in the Interrogation Chamber. "Janine toys nervously with a laptop bag." Janine carries a chocolate biscuit, a laptop bag, and a microfilm. The microfilm is

secret. The laptop bag is wearable. In the bag is a laptop computer.

Rule for deciding the concealed possessions of something: if the particular possession is secret, yes; otherwise no.

Instead of examining someone: say "[The noun] is openly carrying [a list of unconcealed things carried by the noun]."

Test me with "wave wand / examine janine / examine brian".

---

226 **ExampleFour Cheeses** *RB*

A system of telephones on which the player can call distant persons and have conversations.

"Four Cheeses"

Section 1 - Telephones and Connections

A telephone is a kind of thing. Understand "phone" as a telephone.

Understand "call [any telephone] on [something]" as calling it on. Understand "call [any telephone]" as calling it on. Understand the commands "dial" or "phone" or "telephone" as "call". Understand "call [any known

```
person]" as calling it by name on.


Connection relates one thing to another (called the other
party).


The verb to reach means the connection relation.


Calling it on is an action applying to one visible thing
and one thing.


Check calling it on:
    if the second noun is not a telephone, say "[The
second noun] is unlikely to be much use in that respect."
instead;
    if the second noun is the noun, say "You get a busy
signal." instead.


Carry out calling it on:
    if a person (called the listener) can see the noun, now
the player reaches the listener.
```

Because we've said that connection is a reciprocal, one-to-one relationship, Inform will do the rest of the bookkeeping: if (for instance) we telephone someone else, the first connection will be broken automatically.

```
Report calling it on:
    say "'Hello?' says [the other party of the player]."
```

To avoid annoyance, we should also let the player use CALL #### as well as CALL #### ON TELEPHONE. A rule from the chapter on Activities comes in handy here:

```
Rule for supplying a missing second noun while calling
```

```
something on:
    assign a phone.


To assign a phone:
    if the player can touch a telephone (called the current
phone):
        say "(on [the current phone])[line break]";
        now the second noun is the current phone;
    otherwise:
        say "You don't have a phone handy."
```

Things might be a little more complicated if we had cell phones that could be moved around, but for right now the player can only touch a maximum of one phone at a time.

Suppose we further want to allow the player to call people up by name, but only if they've already been encountered or are familiar to the player for some reason.

```
A person can be known or unknown.


Understand "call [any known person]" as calling it by
name on.


Understand "call [any known person] on [something]" as
calling it by name on.


Rule for supplying a missing second noun while calling
something by name on:
    assign a phone.


Calling it by name on is an action applying to one visible
thing and one thing.
```

Check calling it by name on:
    if the noun is in the location, say "[The noun] is right here." instead.

Carry out calling it by name on:
    if the noun can touch a telephone (called the link), try calling the link on the second noun;
    otherwise say "You can't reach [the noun]." instead.

Before calling something on something when the player reaches someone:
    say "(first ending your conversation with [the other party of the player])[command clarification break]";
    end current conversation.

Understand "hang up [something]" as hanging up.

Hanging up is an action applying to one thing.

Check hanging up:
    if the noun is not a telephone, say "You can't hang up [the noun]." instead;
    if the player does not reach someone, say "You're not on the line with anyone." instead.

Carry out hanging up:
    now the player does not reach anyone.

Report hanging up:
    say "You put down [the noun], cutting the connection."

```
Before going somewhere when the player reaches
someone:
    say "(first hanging up on [the other party of the
player])[command clarification break]";
    end current conversation.
```

And finally we want to make sure that calling random other numbers produces a sensible result:

```
Understand "call [text]" as misdialling. Misdialling is an
action applying to one topic. Carry out misdialling: say
"The phone rings and rings but no one answers."


Understand "call 911" or "call 999" or "call police" or
"call fire department" as a mistake ("After strict
warnings, you've given up making prank calls to
emergency services.").


Before misdialling when the player reaches someone:
    say "(first ending your conversation with [the other
party of the player])[command clarification break]";
    end current conversation.


To end current conversation:
    let the current phone be a random telephone which
can be touched by the player;
    silently try hanging up the current phone.


After deciding the scope of the player while the player
reaches someone:
    place the other party of the player in scope, but not its
contents.
```

A note about this scope addition: the player can refer to the other party whenever he has the other person on the phone. He can't, however, see or refer to anything that person might be holding or wearing, thanks to the "but not its contents" option.

Furthermore, the player can't actually do anything to that person that

requires touching. That's because of the reaching inside rules, which govern whether the player can reach through intervening barriers such as rooms. (See the Advanced Actions chapter for more about changing reachability.) There are two things we might want to be careful about, though.

First, we should specifically disallow the player from looking at the person on the other end of the line. Since sight doesn't require touching, the reaching inside rules will not be consulted about a command such as EXAMINE BOSS or LOOK UNDER BOSS. We can, however, intervene in such cases using the visibility rules, which are consulted for any actions that "require light" (including EXAMINE and LOOK UNDER). Here again we borrow some options from the Advanced Actions chapter:

```
To decide whether acting through the line:
    if the noun is something and the location of the noun
is not the location of the player:
        yes;
    if the second noun is something and the location of
the second noun is not the location of the player:
        yes;
    no.


Visibility rule when acting through the line:
    there is insufficient light.


Rule for printing a refusal to act in the dark when acting
through the line:
    say "You're not on a video phone, so you can only
hear." instead.
```

Second, though the existing reaching inside rules are adequate to stop us from touching the person on the other end of the line, the response that's currently printed is a bit generic: it just says "You can't reach into [the room containing the person]." Let's add our own custom reply, instead:

```
A rule for reaching inside a room (called destination):
    if the other party of the player is enclosed by the
destination:
        say "Though you're on the line with [the other party
```

```
of the player], you can't physically reach to [the
destination].";
        deny access.
```

Section 2 - Conversation over the Phone, In General

This portion supplies a simple method of conversation; but we could substitute some completely different conversation system if appropriate. The effect of the telephones is that we are allowed to talk to characters in distant locations under certain circumstances, after which the usual conversation rules apply.

```
Instead of listening to a telephone when the player
reaches someone:
    say "You can hear [the other party of the player]
breathing."
```

```
Before listening to someone when the player cannot
touch the noun:
    say "[The noun] is waiting for you to carry on the
conversation." instead.
```

```
A person has a table name called chatter.
```

```
Before telling someone about something:
    try asking the noun about it instead.
```

```
Before answering someone that something:
    say "Best to confine your conversation to questions
and answers." instead.
```

```
Before asking someone about something:
    if the topic understood is a topic listed in the chatter of
the noun, say "[reply entry][paragraph break]" instead;
```

```
otherwise say "[The noun] does not reply." instead.
```

```
Section 3 - The Scenario
```

```
The Guard House is a room. "Here you spend all your
nights. Bullet-proof windows offer a panoramic view of
serene cliffs, palm trees, and a moonlit ocean.
Occasionally someone is foolish enough to try a cliff
ascent or even an attack by helicopter, but lately things
have been pretty quiet.
```

```
The mansion is up the hill behind you, security lights
ablaze."
```

```
The grey telephone is a telephone in the Guard House.
Understand "6885" as the grey telephone. "Before you
is a grey telephone. In black marker someone has
written on it: MAIN OFFICE 2802."
```

```
Before going a direction in the Guard House, say "And
leave your post? The boss would have you flayed. No
kidding." instead.
```

In a game where the player could walk around, we would of course want to add a before rule so that he automatically hung up any phone he was using before leaving the room.

```
The Main Office is a room. The boss is a known woman
in the Main Office. A telephone called the red telephone
is in the Main Office. Understand "2802" as the red
telephone.
```

```
The Guild is a room. The ninja is an unknown man in
the Guild. A telephone called the black telephone is in
```

the Guild. Understand "4431" as the black telephone.

Potter's Pizza is a room. The pizza delivery boy is a known man in Pizza. A telephone called the saucy telephone is in Pizza. Understand "8885" as the saucy telephone.

The chatter of the boss is the Table of Boss Conversation. The chatter of the delivery boy is the Table of Pizza Conversation. The chatter of the ninja is the Table of Ninja Conversation.

After calling the red telephone on something for the first time:
    say "'Yes?' asks the boss. Her voice is especially husky this evening. Maybe that night of passion isn't so far off after all."

Table of Boss Conversation

topic	reply
"love/passion/night/night"	"'Oh, what a night it was!' she sighs dreamily. 'Oh. Maybe she'd go for some pizza, though."
"pizza"	"'I'd love some. No pepperoni, though,' she says, sounding dreamy. Yes, this is definitely time for a call to your old friend, the pizza boy."
"ninja/attack"	"'Don't worry about it,' says the boss crisply. 'I have everything under control.'"

Table of Pizza Conversation

topic	reply
"pizza"	"'Pepperoni special tonight!' he says proudly."
"pepperoni"	"'Pepperoni is included free on ALL our pizzas,' he says proudly."
"no pepperoni"	"'Well, I don't see why you'd want that,' replies the boy sniffily. 'It's free!'"
"jalapeno"	"'Sorry, we're out of jalapenos this evening. There was a run on them.'"

"Sausage, sure, we can do you sausage.'"
"There's currently an embargo on Canadian pig products.'"
"We've four kinds,' says the boy, then lowers his voice confidentially. 'Actually, two of them are the same. Nobody ever counts. The stringy one, the one that comes in dollops and the orangey one. You know.'"
"We can put pineapple on there, sure,' says the delivery boy, in a tone that lets you know his opinion of people who order fruit-based pizzas."
"What kind of combination is that?' demands the delivery boy, finally losing all self-control."
"Well, I don't know,' says the boy in a worried voice. 'Last time I came there were attack dogs. And ninjas.'"
"There's no way with a big tip you don't live to spend,' says the delivery boy quite firmly."

Table of Ninja Conversation

reply
"Yes, still attack tonight,' confirms the voice at the other end of the line."
"The voice, in tones of velvet, indicates that it cannot guarantee the safety of any delivery persons whomsoever."

Test me with "call 2802 / examine boss / ask boss about night of passion / ask boss about pizza / listen to telephone / call delivery boy on telephone / ask boy about cheese / tell boy about no pepperoni / ask boy about delivery / tell boy about massive gratuity / attack boy".

...and it more or less writes itself from there.

---

227 **ExampleTransmutations** *RB*

A machine that turns objects into other, similar objects.

Suppose we want to have a machine in our game that can transmute one item into another, similar object with different properties: a bag of jelly

beans into a bag of jewels, for instance. Thus each item will be associated with some number of equivalents -- the other objects it can turn into. This is a handy use for group relations:

```
"Transmutations"

Workshop is a room.

Transmutation relates things to each other in groups.
The verb to become means the transmutation relation.

Definition: a thing is transmutable if it becomes more
than one thing. [* It always becomes itself.]

A thing can be valuable. Something valuable called a
bag of jewels is carried by the player. It becomes the
bag of gunpowder and the bag of jelly beans.

A thing can be dangerous. The bag of gunpowder is a
dangerous thing.

The bag of jelly beans is an edible thing.

The machine is fixed in place in the workshop.

The can't insert into what's not a container rule does
nothing when inserting something into the machine.
```

```
Check inserting something which is not transmutable
into the machine:
    instead say "You can't transmute that."


To decide which thing is new form of (obj - edible thing):
decide on a random valuable thing which becomes obj.


To decide which thing is new form of (obj - dangerous
thing): decide on a random edible thing which becomes
obj.


To decide which thing is new form of (obj - valuable
thing): decide on a random dangerous thing which
becomes obj.


Carry out inserting something into the machine:
    now the noun is nowhere;
    now the player carries the new form of the noun.


Report inserting something edible into the machine:
    say "The machine clicks, whirrs, and spits out [a new
form of the noun]. You're rich!";
    rule succeeds.


Report inserting something dangerous into the machine:
    say "The machine clicks, whirrs, and in a shower of
flavor crystals, spits out [a new form of the noun].";
    rule succeeds.


Report inserting something valuable into the machine:
    say "The machine clicks, whirrs, and with a violent
roar, spits out [a new form of the noun].";
    rule succeeds.
```

```
Test me with "i / put jewels in machine / i / put
gunpowder in machine / i / put beans in machine".
```

In this example we have only defined a single set of transmutable objects, but we could easily expand to include other groups.

(Thanks to Jesse McGrew for proposing this example.)

---

228 ExampleOtranto

RB

A kind of rope which can be tied to objects and used to anchor the player or drag items from room to room.

The range of things one might want to do with a rope in a work of interactive fiction is fairly overwhelming. One might, in theory, swing from ropes; use them to tie containers shut; cut them up into smaller ropes; tie them together into longer ropes; employ them as fuses; bind other characters with them, or the player character.

Our rope implementation is, by these lights, reasonably simple, but it does account for the possibility of tying and untying both ends; using ropes to descend into lower rooms; pulling objects tied to the far end of the rope; and dragging objects from place to place.

```
"Otranto"
```

We start by coming up with a rope.

```
A rope is a kind of thing.
```

```
Definition: a thing is nonrope if it is not a rope. [The
perfect idiocy of this statement notwithstanding, having
a shortcut will come in very handy later]
```

```
Attachment relates things to each other in groups. The
```

```
verb to be stuck to means the attachment relation.

Definition: a thing is tied if the number of things stuck to
it is greater than 1.

Definition: a thing is free if it is not tied.

Definition: a rope is free if the number of nonrope things
stuck to it is less than 2.

Definition: a thing is hindering if it is stuck to the noun
and it is not within the location.

A thing can be round or unevenly shaped. A thing is
usually round.

Definition: something is anchored if it is fixed in place or
it is scenery or it is part of an anchored thing.

Definition: something is draggable if it is not had by the
player and it is not the player and it is not anchored.
```

Now, we want a rope to be described in terms of the way it is tied, when it's described in a room description.

```
Rule for writing a paragraph about a rope (called the
coil):
    if the coil is stuck to something which is in a room
(called the next room) which is not the location:
        let the way be the best route from the location to
the next room;
        if the way is up or the way is down:
```

```
            say "[The coil] runs [way] into [the next room].";
        otherwise:
            say "[The coil] snakes across the floor [way]
towards [the next room].";
    otherwise:
        say "There is [a coil] here[if the coil is stuck to a
visible nonrope thing], tied to [the list of nonrope visible
things which are stuck to the coil][end if]."



To decide what room is the home of (item - a thing):
    if item is a door:
        let front cut be the number of moves from the
location to the front side of the item;
        let back cut be the number of moves from the
location to the back side of the item;
        if front cut is -1, let front cut be 999;
        if back cut is -1, let back cut be 999;
        if the location encloses the item, decide on the
location;
        if front cut is greater than back cut, decide on the
back side of the item;
        decide on the front side of the item;
    decide on the location of the item.



Rule for writing a paragraph about a nonrope thing
(called the anchor) which is stuck to a rope (called the
coil):
    if the coil is in an adjacent room:
        let the next room be the home of the coil;
        let the way be the best route from the location to
the next room;
        if the way is up or the way is down:
            say "[The coil] runs [way] from [the anchor] into
[the next room].";
        otherwise:
            say "From [the anchor] runs [a coil], heading off
toward [the way].";
    otherwise:
        if the coil is stuck to something which is not visible,
            say "[The coil] is tied to [the anchor][if the coil is
stuck to something in an adjacent room (called the next
room)], and from there runs off towards [the next room]
[end if]."
```

We need a way to account for it when it's being carried, as well.

```
After printing the name of a rope (called the tied object)
while taking inventory:
    if something nonrope is stuck to the tied object:
        say " (attached to [the list of nonrope things which
are stuck to the tied object])";
    otherwise:
        say " (with both ends free)".
```

And, indeed, whenever the player examines a rope, we should see what's connected.

```
Instead of examining a rope (called the cord) when
something is stuck to the cord:
    say "[The noun] is tied to [the list of secondary things
which are stuck to the noun]."
```

Similarly, any time the player looks at something tied to a rope.

```
After examining the player when the player is stuck to
something which is not the player:
    say "You're currently lashed to [the list of secondary
things stuck to the noun]."
```

```
After examining something which is stuck to something
secondary:
    say "[The noun] is currently attached to [the list of
secondary things stuck to the noun]."
```

We also need to make sure that the rope can be interacted with properly even when it's partly in the next room.

```
After deciding the scope of the player:
    if something stuck to a rope (called the coil) is in the
location, place the coil in scope.
```

```
A reaching inside rule:
    if the noun is a rope:
```

```
        let the anchor be a random visible thing stuck to
    the noun;
        if the anchor is touchable, allow access.
```

Now tying:

```
Before tying something to a rope:
    if the noun is stuck to the second noun, say "[The
noun] and [the second noun] are already tied together."
instead;
    if the second noun is not free, say "[The second
noun] has no ends free." instead;
    if the noun is round, say "You can't realistically tie
anything to [the noun]." instead.

Instead of tying a rope to something:
    try tying the second noun to the noun.

Instead of tying something to a rope:
    now the noun is stuck to the second noun;
    say "You loop [the second noun] around [the noun]
and knot firmly."

Instead of tying something to a nonrope tied thing:
    let the coil be a random rope stuck to the second
noun;
    try tying the noun to the coil.

Instead of tying a nonrope tied thing to something:
    let the coil be a random rope stuck to the noun;
    try tying the second noun to the coil.

Instead of tying a free nonrope thing to a free nonrope
thing:
    if the player carries a free rope (called the coil):
        try tying the noun to the coil;
        if the noun is stuck to the coil and the coil is free:
```

```
            try tying the second noun to the coil;
    otherwise:
        say "You lack the requisite spare rope."


Understand "untie [something] from [something]" as
untying it from. Understand "untie [something]" as
untying it from.


Rule for supplying a missing second noun while untying
something from:
    if the number of secondary things stuck to the noun is
0, say "[The noun] is already entirely free." instead;
    if the noun is a rope:
        if the number of touchable nonrope things which
are stuck to the noun > 1:
            say "You'll have to say which thing you want to
untie [the noun] from.";
            rule fails;
        otherwise:
            if the number of touchable nonrope things stuck
to the noun is 0, say "You can't reach [the random
nonrope thing stuck to the noun]." instead;
            let the tied object be a random touchable
nonrope thing which is stuck to the noun;
            say "(from [the tied object])[line break]";
            now the second noun is the tied object;
    otherwise:
        if the noun is stuck to a rope (called the tied
object):
            say "(from [the tied object])[line break]";
            now the second noun is the tied object.


Untying it from is an action applying to two things.


Before untying a rope from something: try untying the
second noun from the noun instead.


Before untying something from a rope:
    if the second noun is not held:
```

```
        say "(first picking up [the second noun])[line
    break]";
        try taking the second noun.


    Check untying it from:
        unless the noun is stuck to the second noun or the
    second noun is stuck to the noun,
        say "[The noun] and [the second noun] are already
    not tied together." instead.


    Carry out untying it from:
        now the noun is not stuck to the second noun.


    Report untying it from:
        say "Untied."
```

Another part of the fun of a rope is that you can drag things from another room:

```
    After reading a command: now every thing is
    unmentioned.


    Before pulling something anchored: say "[The noun] is
    firmly anchored." instead.


    Instead of pulling something tied:
        if the noun is unmentioned:
            say "The impulse is transmitted to [the list of
    pullable things stuck to the noun].";
            repeat with item running through pullable things
    stuck to the noun:
                say "[item]: [run paragraph on]";
                try pulling the item;
        if the noun is a rope and the noun is not within the
    location:
            if the number of nonrope hindering things is 0,
```

```
        move the noun to the location;
    otherwise:
        continue the action.


Before pulling something which is not visible:
    if the noun is anchored:
        say "[The noun] resists, for whatever reason."
instead;
    otherwise:
        let space be the holder of the noun;
        let way be the best route from the space to the
location;
        if the way is a direction:
            move the noun to the location;
            say "[The noun] [if the way is
up]rises[otherwise]slides[end if] into view." instead;
        otherwise:
            move the noun to the location;
            say "[The noun] slides into view." instead.


Definition: a thing is secondary if it is not the noun.
Definition: a thing is pullable if it is not the noun and it is
not the player.
```

A player who is tied to things should also have some restrictions on his ability to move.

```
Before going a direction (called the way) when the
player has something (called the link) which is stuck to
something anchored (called the anchor):
    let the next room be the home of the anchor;
    if the next room is not a room, continue the action;
    if the next room is the location:
        if the link is stuck to at least two anchored things,
            say "You can't go far while you're carrying [the
link] tied to [the list of anchored things stuck to the link]."
instead;
    otherwise:
        let the safe way be the best route from the location
to the next room;
        if the safe way is the way:
            if the player is not stuck to the anchor, say
"(coiling up your rope again as you go...)";
        otherwise:
```

```
            if the safe way is a direction,
                say "While you have [the link] you can't really
head any direction but [best route from the location to
the next room]." instead;
            otherwise say "You're tied up here." instead.
```

```
Before going a direction (called the way) when the
player is stuck to something anchored (called the
anchor):
    let the next room be the home of the anchor;
    if the next room is not a room, continue the action;
    if the next room is the location:
        if the player is stuck to at least two anchored
things,
            say "You can't go far while you're tied to [the list
of anchored things stuck to the player]." instead;
    otherwise:
        if the best route from the location to the next room
is the way:
            say "(coiling up your rope again as you go...)";
        otherwise:
            say "Your attachments prevent you going any
way but [best route from the location to the next room]."
instead.
```

Sometimes, if the player is tied to a movable object, the moved object will move with him.

```
After going somewhere when the player has something
(called the link) which is stuck to something draggable:
    if the player is not stuck to the link:
        say "You drag along behind you [the list of
draggable things which are stuck to the link].";
        now every draggable thing which is stuck to the link
is in the location;
    continue the action.
```

```
Report going somewhere when the player is stuck to
something draggable:
    say "You drag along behind you [the list of draggable
things which are stuck to the player].";
    now every draggable thing which is stuck to the
player is in the location.
```

And now the actual game and puzzles.

Use full-length room descriptions.

The Fallow Field is a room. "The very land is gloomy, the earth plowed into untended rows that yield no fruit, shadowed by the castle to the north. A chasm, no doubt the product of some upheaval of the earth, opens before your feet.". An oak stump is fixed in place in the Field. "From an oak stump, a few hopeful shoots grow." A hempen rope is a rope in the field. It is stuck to the oak stump and the wooden chest. The stump is unevenly shaped.

The Chasm is below the Field. "Your person is most uncomfortably pressed on every side by the closeness of the walls; to which you may add as a further inconvenience, that the irregularity of the floor making it difficult to walk upright." An iron key is in the Chasm. "An iron key nestles in the cleft of earth, its age indicated by its implausibly great size."

The wooden chest is a unevenly shaped closed openable container in the Chasm. The description of the wooden chest is "A handsome, solid case not long committed to its dank enclosure, or it would long since have rotted." Rule for printing the name of the wooden chest when the chest is not handled: say "deadweight". Understand "dead" or "weight" or "deadweight" as the chest. Before pulling the wooden chest: now the chest is handled.

In the chest is a heavy dagger. The description of the dagger is "Set with red jewels and of a wicked aspect."

Before going down from the Field when the player is not stuck to something anchored:

say "You don't quite dare simply leap into the darkness without some anchor." instead.

Before going down from the Field:
let anchor be a random anchored thing which is stuck to the player;
say "You lower yourself gingerly, hoping that [the anchor] holds your weight..."

Before going up from the Chasm:
if the player cannot touch a rope which is stuck to an anchored thing which is in the Field, say "And how, precisely, do you mean to do that?" instead.

The Castle Hall is north of the Field. "All is desolate: the great hall has no roof, nor is there any glass in the windows. A staircase without banister ascends inside the wall to a musician's gallery without song."

The Musician's Gallery is above the Castle Hall. "Of its former cheery aspect only this remains to the Gallery: that chevrons of red and yellow are painted on the wall. But as these are streaked with rain and grime, the banister pulled away, the roof open to the sky, and the corners made a nesting place for birds, the consolation thereby afforded is but slight."

The pointed door is north of the Musician's Gallery and south of the Sinister Attic. It is a closed locked openable door. "A pointed door of particularly grim and uninviting aspect leads north." The pointed door is lockable and unevenly shaped. The description of the pointed door is "A door coming to a gothic point and fitted with iron fittings of great strength. The handle looks particularly well-attached." The iron key unlocks the pointed door.

A rule for reaching inside the Musician's Gallery:
    allow access.

A rule for reaching inside the Sinister Attic:
    allow access.

Instead of opening the pointed door for the first time:
    say "When you rattle at the door, there arises from beyond a terrible shrill noise as though something beyond exults in its imminent release."

After opening the trapped pointed door when the player can see the pointed door:
    say "Thousands of bats fly from the pointed door, attacking you!";
    end the story.

After opening the pointed door when the player cannot see the pointed door: now the pointed door is untrapped; continue the action.

The pointed door can be trapped or untrapped. The pointed door is trapped.

Before pulling the pointed door: try opening the pointed door instead.

The player is unevenly shaped.

After going to the sinister attic:
    say "You have arrived at the goal of your quest!";
    end the story finally.

Test me with "x rope / pull rope / get chest / untie rope from chest / tie rope to me / down / get key / up / untie rope from stump / north / up / unlock pointed door with key / open it / tie rope to door / down / pull rope / up / north".

Test death with "x rope / pull rope / get chest / untie rope from chest / tie rope to me / down / get key / up / untie rope from stump / north / up / unlock pointed door with key / open it / g"

---

229 **Example** **Unthinkable Alliances** *RB*

People are to be grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give notice of quitting, and to become a lone wolf.

The following is best tested by experimentally kissing and/or attacking, and typing RELATIONS after every change to see the effect.

"Unthinkable Alliances"

Unthinkable Solutions is a room. Sophie, Daisy, Ryan and Owen are in Unthinkable Solutions.

Alliance relates people to each other in groups. The verb to help means the alliance relation.

Sophie helps Ryan. Daisy helps Ryan. Owen helps the player.

```
Instead of kissing someone (called the blessed one):
    say "Smack!";
    now the player helps the blessed one.

Instead of attacking someone (called the vilified one):
    say "Smack!";
    now the player does not help the vilified one.

Test me with "relations / kiss sophie / relations / hit ryan
/ relations".
```

---

230 **Example** **The Unexamined Life** *RB*

An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a nudge he wants at any given moment.

Hint systems in IF come in a variety of flavors: some are a static, prewritten set of guidelines (which might exist in a menu or outside the game entirely); others are built in as part of the program, and attempt to adapt to the situation the player currently faces. Adaptive hints have the advantage that they are less likely to reveal information for which the player is not ready, and the disadvantage that they are more work for the author.

The exercise here is to write an adaptive hint system that will both respond in agile ways to the state of the world model and require a minimum of authorial fussing. We also want the player to be able to ask for a hint about any object he encounters in the game world: this will let him be specific and avoid accidentally receiving hints about the wrong puzzles.

Our baseline assumption is that a player may find a puzzle unsolvable for one of two reasons: he either hasn't seen the relevant clue, or he hasn't got the relevant equipment. If these are true, then he should be given hints about how to find this information, and then once he has it, more specific hints about the puzzle itself -- ending, as a last resort, with the exact command(s) he will need to use in order to bring about the solution.

In practice, there are other possibilities, but this will do for an example.

We begin by defining our relations:

```
"The Unexamined Life"

Use scoring.

Explaining relates one thing to various things. The verb
to explain means the explaining relation.

Instead of hinting about something when something
unexamined (called the clue) explains the noun:
    say "You're still missing some information that might
be useful to understanding the problem. [More]";
    if player consents, try hinting about the clue.

Requiring relates one thing to various things. The verb
to require means the requiring relation.

Instead of hinting about something when the noun
requires something (called the implement) which is not
carried by the player:
    say "You're missing an object that might be useful to
resolving this problem. [More]";
    if player consents, try hinting about the implement.

Hinting about is an action applying to one visible thing.
Understand "hint about [any thing]" as hinting about.
```

This allows us to create the most absolutely generic sort of hint -- boring, perhaps, but in practice the player often just needs a nudge about what part of the game world he should be examining for a solution:

```
Carry out hinting about:
    if something explains the noun, say "You might want
to review [the list of things which explain the noun]. ";
    if the noun requires something:
        say "You should be sure that you have [the list of
things required by the noun]. ";
    otherwise:
        say "Sorry, I can't advise you further on that.".
```

These things cover hinting about objects that are themselves puzzles. But what if the player asks for hints about a tool or piece of information because he doesn't know how to apply it yet? We might want to give some guidance there, as well.

```
Carry out hinting about something which explains
something (called target):
    if target is unseen, say "[The noun] might prove
useful information, sooner or later." instead;
    otherwise say "You could examine [the noun]."
instead.
```

```
Carry out hinting about something which is required by
something:
    say "[The noun] might be useful to have. [More]";
    if player consents:
        if a seen thing requires the noun, say "[The noun]
may help with [the list of seen things which require the
noun]." instead;
        otherwise say "There are [number of things which
require the noun in words] problems for which [the noun]
might come in handy." instead.
```

Now we have these general hints written, but we want to pre-empt them if the player has not yet fulfilled all the prerequisites.

```
Instead of hinting about something unseen:
    if the noun is visible:
        now the noun is seen;
        continue the action;
    say "Perhaps you should explore further. ";
    if the ultimate location of the noun is an unvisited
```

```
room:
        try hinting about the ultimate location of the noun;
    otherwise:
        if the ultimate location of the noun is the location:
            say "You're in the correct room right now[if the
visible shell of the noun is a thing]. Try further exploring
[the visible shell of the noun][end if].";
        otherwise:
            try hinting about the ultimate location of the
noun.



Instead of hinting about a visited room:
    say "There's a room you've visited, but you haven't
exhausted all there is to see there. [More]";
    if player consents:
        say "Try going back to [the noun]. [More]";
        if player consents, direct player to the noun.



Instead of hinting about an unvisited room:
    say "There's a room you haven't yet visited. [More]";
    if player consents, direct player to the noun.



To direct player to (goal - a room):
    let way be the best route from location to the goal,
using even locked doors;
    if way is a direction, say "Try going [way] to start your
explorations.";
    otherwise say "Sorry, the route is an indirect one.".



Instead of hinting about a portable seen thing which is
not visible:
    if the noun is scenery, continue the action;
    say "You have seen the item you need to solve this
problem, but it's not in sight at the moment. [More]";
    if player consents:
        try hinting about the ultimate location of the noun.
```

And this business of "seen" things requires, of course, that we keep track:

```
A thing can be seen or unseen. A thing is usually
unseen. The player is seen. After printing the name of
something (called target): now the target is seen.
```

That "After printing..." rule means that as soon as the game automatically prints the name of an object, it tags that object as having been "seen" by the player. This requires just a little care on our part, that we never mention an object without using the game's printing rules. Still, it is much easier than most other possible forms of bookkeeping.

We also need to deal with the question of whether the player has examined an object, for those objects whose descriptions carry vital information:

```
A thing can be examined or unexamined. A thing is
usually unexamined. Carry out examining something:
now the noun is examined.
```

In practice, there might be other ways of getting vital facts, and in a more sophisticated puzzle game we might need a more sophisticated model to track this. But examined or unexamined will do for now.

```
To decide what room is the ultimate location of (item - a
thing):
    let place be the holder of the item;
    while the place is a thing:
        let the place be the holder of the place;
    if the place is a room, decide on the place.
```

```
To decide what thing is the visible shell of (item - a
thing):
    if item is visible, decide on the item;
    let place be the holder of the item;
    while place is a thing and place is not visible:
        let place be the holder of the place;
    if the place is visible, decide on the place.
```

```
To say more:
    say "[paragraph break]Shall I go on? > ".
```

That covers most of the generic hints, but let's also add some slightly

more precise hints about a few kinds of objects that are especially important in the model world. These hints will probably not be very interesting to a seasoned IF veteran, but a novice player who does not know the wording or cannot guess what something might be for may still find them useful:

```
Carry out hinting about a locked lockable thing:
    say "You could unlock [the noun] with [the matching
key of the noun]." instead.


Instead of hinting about a locked thing when the
matching key of the noun is not carried by the player:
    if the player can see the matching key of the noun:
        say "Perhaps [the matching key of the noun] would
help.";
    otherwise:
        say "[The noun] is locked. There must be a key
around somewhere. [More]";
        if player consents, try hinting about the matching
key of the noun.


Carry out hinting about a closed openable unlocked
thing:
    say "You could open [the noun]." instead.


Carry out hinting about an open door:
    say "You could enter [the noun]." instead.


Carry out hinting about an unexamined thing:
    say "You might find out something if you examine [the
noun]." instead.


Carry out hinting about an edible thing:
    say "You could eat [the noun]." instead.
```

```
Carry out hinting about a wearable thing:
    say "You could wear [the noun]." instead.
```

```
Carry out hinting about a pushable between rooms
thing:
    say "You could push [the noun] some direction."
instead.
```

Now to the actual objects in the game:

```
The Crypt is a room. "This squat, barrel-vaulted
chamber runs roughly north-south. Along either side are
the graves of Saxon kings and early bishops of the
church long since gone to dust -- one [tomb] in particular
looks undisturbed."
```

Notice that we used the bracketed tomb here: the tomb is scenery, and if we do not use the name-printing function, Inform will not register that we have mentioned it to the player.

```
The tomb is scenery in the Crypt. The tomb is openable
and closed. The silver dagger is a thing in the tomb.
Understand "tombs" as the tomb. The description of the
silver dagger is "Gleaming in a soft light all its own. Its
blade is figured with running deer and its hilt is made of
horn." The wight requires the silver dagger. The tomb
requires the pry bar.
```

```
Instead of opening the tomb when the player does not
carry the pry bar:
    say "The lids are stone, too heavy for you to raise
without some implement."
```

Now we can add specific hints to replace the generic ones:

```
Carry out hinting about the tomb:
    say "The lids are heavy, but you can open them when
you carry the pry bar."
```

The rest of the hint system ensures that the player will not see this final suggestion until he has the pry bar, since the tomb "requires" the pry bar. Having the hint there doesn't excuse us from providing some alternate wording in case the player solves this not-very-difficult conundrum on his own, though:

```
Understand "pry [something] with [something preferably
held]" as unlocking it with. Understand the commands
"lever" or "prise" as "pry".


Instead of unlocking something with the pry bar, try
opening the noun.


The wight is a man in the Crypt. "[The wight] lurks near
the south exit." The description of wight is "Old English
[italic type]wiht[roman type]: a thing, a creature. It is little
more than the memory of a life ill-lived, but it lingers
here." Understand "wiht" or "creature" or "ghost" as the
wight.


Instead of going south in the presence of wight:
    say "The wight breathes chill into your face.


Your head swims, and you are aware that you no longer
have the willpower to go in that direction."


Fresh Air is south from the Crypt.


After going to Fresh Air:
    increment the score;
    say "Congratulations, you have escaped!";
    end the story finally.
```

The inscription is fixed in place in the Crypt. "Someone has painstakingly carved [an inscription] into the wall above the door." The description is "Squinting, you decipher the Latin text: [italic type]Silver causes harm to those that live though dead[roman type]." The inscription explains wight.

The Treasure Chamber is north of the Crypt. "The walls are thick, the high windows promisingly barred with iron. But for all this there is no hint of any valuable stores remaining."

The pry bar is in the Treasure Chamber. "One of the window bars, rusted from its place, lies in a puddle of water." Understand "window" or "bars" as the pry bar. The description of the pry bar is "A few feet long, and not entirely rusted into uselessness yet."

Instead of giving the dagger to wight:
    say "The wight recoils, appalled."

Carry out hinting about wight:
    say "You will have to find some way to get wight to come in physical contact with the silver dagger, which he will certainly not do willingly. [More]";
    if player consents, say "You could, for instance, throw it at him." instead;
    otherwise stop the action.

Understand "touch [something] with [something]" as putting it on (with nouns reversed). Understand "hit [someone] with [something]" as putting it on (with nouns reversed).

```
Instead of attacking the wight:
    say "You can't force yourself to approach close
enough for hand to hand combat: if, indeed, the wight
has hands."


Instead of putting the dagger on wight:
    say "The wight fades out of your way without ever
coming into contact with the dagger. Perhaps a more
projectile method would work better."


Instead of putting something on wight:
    say "The wight dodges you."


Instead of throwing the dagger at wight:
    now the wight is nowhere;
    move the dagger to the location;
    increment the score;
    say "The dagger passes through its airy form with a
rending like the rip of silk. The fragments dissipate at
once."


The maximum score is 2.


Test me with "hint about wight / north / get bar / south /
open tomb / get dagger / south / hint about wight / read
inscription / hint about wight / attack wight / throw
dagger at wight / south".
```

Note that, if using TEST ME to run through the solution on the Z-machine, we will have to answer a few yes/no questions along the way.

For Glulx, the code should instead read something like

```
Test me with "hint about wight / y / north / get bar /
south / open tomb / get dagger / south / hint about wight
/ y / read inscription / hint about wight / y / attack wight /
throw dagger at wight / south".
```

231 **Example The Abolition of Love** *RB*

A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

Suppose we are modeling a complex society seething with interpersonal relations of every kind.

"The Abolition of Love"

Section 1 - Relation types

Loving relates one person to one person.

Noticing relates various people to one person.

Impressing relates one person to various people.

Fancying relates various people to various people.

Acquaintance relates people to each other.

Marriage relates one person to another.

Alliance relates people to each other in groups.

The Chapel is a room. Elizabeth, Wickham and Darcy are people in the Chapel. Mr Bennett and Mrs Bennett are people in the Chapel. Georgiana is a person in the Chapel.

The verb to love means the loving relation.

The verb to notice means the noticing relation.

The verb to impress means the impressing relation.

The verb to fancy means the fancying relation.

The verb to know means the acquaintance relation.

The verb to be married to means the marriage relation.

The verb to be related to means the alliance relation.

Elizabeth loves Darcy. Elizabeth fancies Darcy. Elizabeth notices Darcy. Elizabeth impresses Darcy.

Mr Bennett is related to Mrs Bennett and Elizabeth. Mr Bennett is married to Mrs Bennett.

```
Georgiana is related to Darcy.
```

Now we want ways to set and unset all of these relations. (In the interests of thoroughness, we may get a bit far-fetched here. It is not recommended in practice that we make the player guess the verb "traduce".)

```
Section 2 - Setting and Unsetting Love (1-1)

Understand "infatuate [someone] with [someone]" as
infatuating it with. Infatuating it with is an action applying
to two visible things.

Carry out infatuating it with:
    now the noun loves the second noun.

Report infatuating it with:
    say "Now [the noun] loves [a random person loved by
the noun][if the second noun loves someone], while [the
second noun] loves [a random person loved by the
second noun][end if]."

Understand "embitter [someone] toward [someone]" as
embittering it toward. Embittering it toward is an action
applying to two visible things.

Carry out embittering it toward:
    now the noun does not love the second noun.

Report embittering it toward:
    say "[The noun] sees [the second noun] in a different
light and no longer feels any affection."
```

Because love is a 1-1 relation, a person cannot love more than one other

character at a time. Whenever we set a character to love a new person, that person ceases to love the character loved before. It is a fickle world.

One to various relations are a bit more open: we can say someone impresses multiple other characters, and our additions to the list do not override the initial ones.

```
Section 3 - Setting and Unsetting Impressed (1-V)


Understand "commend [someone] to [someone]" as
commending it to. Commending it to is an action
applying to two visible things.


Carry out commending it to:
    now the noun impresses the second noun.


Report commending it to:
    say "[The second noun] takes a very decided interest
in [the noun]."


Understand "traduce [someone] to [someone]" as
traducing it to. Traducing it to is an action applying to
two visible things.


Carry out traducing it to:
    now the noun does not impress the second noun.


Report traducing it to:
    say "[The second noun], hearing your story, decides
not to be at all impressed with [the noun]."
```

And because this is a one-to-various relation, we can also make statements which set multiple relations at once, so:

```
Understand "celebrate [someone]" as celebrating.
Celebrating is an action applying to one visible thing.

Carry out celebrating:
    now the noun impresses every person.

Report celebrating:
    say "[The list of people who are impressed by the
noun] take a very decided interest in [the noun]."

Understand "slander [someone]" as slandering to.
Slandering to is an action applying to one visible thing.

Carry out slandering to:
    now every person is not impressed by the noun.

Report slandering to:
    say "Now [the noun] impresses [the list of people who
are impressed by the noun]."
```

Note that the above unsetting is not equivalent to "now the noun does not impress every person" -- which would be ambiguous in spoken English, as well. Various-to-one relations are similar:

```
Section 4 - Setting and Unsetting Noticing (V-1)

Understand "draw the attention of [someone] to
[someone]" as drawing the attention of it to. Drawing the
attention of it to is an action applying to two visible
things.
```

Carry out drawing the attention of it to:
    now the noun notices the second noun.

Report drawing the attention of it to:
    say "[The noun] glances thoughtfully in the direction of [the second noun]."

Understand "distract [someone] from [someone]" as distracting it from. Distracting it from is an action applying to two visible things.

Carry out distracting it from:
    now the noun does not notice the second noun.

Report distracting it from:
    say "You distract [the noun] from [the second noun]."

Understand "draw attention to [someone]" as drawing attention to. Drawing attention to is an action applying to one visible thing.

Carry out drawing attention to:
    now every person notices the noun.

Report drawing attention to:
    say "You quickly cause everyone to attend to [the noun]."

Understand "outshine [someone]" as outshining.

Outshining is an action applying to one visible thing.

Carry out outshining:
    now every person does not notice the noun.

Report outshining:
    say "You quickly distract everyone from [the noun]."

Section 5 - Setting and Unsetting Fancying (V-V)

Understand "flatter [someone]" as flattering. Flattering is an action applying to one thing.

Carry out flattering:
    now every person fancies the noun.

Report flattering:
    say "You draw down universal admiration for [the noun] by casting him or her in a flattering light."

Understand "unflatter [someone]" as unflattering. [Okay, okay, but it's four am.] Unflattering is an action applying to one thing.

Carry out unflattering:
    now every person does not fancy the noun.

Report unflattering:
    say "[The noun] gives everyone a universal disgust."

Understand "admire [someone]" as admiring. Admiring is an action applying to one thing.

Carry out admiring: now the player fancies the noun.

Report admiring: say "You find you rather fancy [the noun]."

Understand "loathe [someone]" as loathing. Loathing is an action applying to one thing.

Carry out loathing: now the player does not fancy the noun.

Report loathing: say "You take [the noun] in disgust."

Understand "cause chaos" as causing chaos. Causing chaos is an action applying to nothing.

Carry out causing chaos:
    now every person fancies every person.

Report causing chaos: say "Now everyone fancies everyone else, which is quite an inconvenient state of affairs."

Understand "relieve chaos" as relieving chaos.

```
Relieving chaos is an action applying to nothing.

Carry out relieving chaos:
    now every person is fancied by no one.

Report relieving chaos: say "Now no one fancies
anyone, which is safe but tedious."
```

Our options for setting and unsetting symmetrical relations are more limited again:

```
Section 6 - Setting and Unsetting Marriage (1-1
Symmetrical)

Understand "marry [someone] to [someone]" as uniting
it in matrimony with. Uniting it in matrimony with is an
action applying to two visible things.

Carry out uniting it in matrimony with:
    now the noun is married to the second noun.

Report uniting it in matrimony with:
    say "You perform the marriage of [the noun] to [the
second noun], joining them to the family of [a list of
people related to the noun]."

Understand "divorce [someone] from [someone]" as
divorcing it from. Divorcing it from is an action applying
to two visible things.

Check divorcing it from:
    if the noun is not married to the second noun, say
```

```
"[The noun] is not married to [the second noun]
anyway." instead.


Carry out divorcing it from:
    now the noun is not married to the second noun.


Report divorcing it from:
    say "[The noun] is now not married to [the second
noun]."
```

When we unset the symmetrical relation on one side, it is automatically set or unset on the other. It is not necessary to say "the second noun is married to the noun" or "the second noun is not married to the noun", even though that is the case.

```
Section 7 - Setting and Unsetting Acquaintance (V-V
Symmetrical)


Understand "introduce [someone] to [someone]" as
introducing it to. Introducing it to is an action applying to
two visible things.


Carry out introducing it to:
    now the noun knows the second noun.


Report introducing it to:
    say "You introduce [the noun] to [the second noun].
Now [the noun] is acquainted with [the list of people who
are known by the noun], and [the second noun] is
acquainted with [the list of people who are known by the
second noun]."


Understand "announce [someone]" as announcing.
Announcing is an action applying to one visible thing.
```

```
Carry out announcing:
    now every person knows the noun.

Report announcing:
    say "You announce [the noun] to the whole
assembled company."

Understand "ostracise [someone]" as ostracising.
Ostracising is an action applying to one visible thing.

Carry out ostracising:
    now every person does not know the noun.

Report ostracising:
    say "You cause everyone present to forget and
pretend not to be acquainted at all with [the noun]."
```

And finally, setting groups:

```
Section 8 - Setting and Unsetting Familial Relations
(Groups)

Understand "make [someone] adopt [someone]" as
forcing it to adopt. Forcing it to adopt is an action
applying to two visible things.

Carry out forcing it to adopt:
    now the noun is related to the second noun.
```

```
Report forcing it to adopt:
    say "Now [the second noun] is related to [the list of
people related to the second noun]."


Understand "make [someone] disown [someone]" as
forcing it to disown. Forcing it to disown is an action
applying to two visible things.


Carry out forcing it to disown:
    now the second noun is not related to the noun.


Report forcing it to disown:
    say "Now [the second noun] is related to [the list of
people who are related to the second noun], and [the
noun] is related to [the list of people who are related to
the noun]."
```

Notice that when we say "the second noun is not related", we remove that person from the group: they are now in a separate group of their own, while the rest of the group's members remain related to one another.

And finally, a long litany of test cases, complete with the relations lists:

```
Test acquaintance with "relations / introduce darcy to
elizabeth / introduce darcy to wickham / announce mr
bennett / relations / ostracise wickham / introduce
georgiana to wickham / relations".


Test impression with "commend georgiana to elizabeth /
relations / celebrate Mrs bennett / relations / traduce
mrs bennett to darcy / relations / slander mrs bennett /
relations".


Test notice with "draw the attention of darcy to elizabeth
/ relations / draw attention to mr bennett / relations /
```

distract darcy from mr bennett / relations / outshine mr bennett / relations".

Test love with "embitter elizabeth toward darcy / relations / infatuate elizabeth with wickham / relations".

Test marriage with "marry elizabeth to darcy / relations / divorce elizabeth from darcy / relations".

Test alliance with "make mr bennett adopt georgiana / relations / make mrs bennett disown georgiana / relations".

Test fancying with "admire elizabeth / relations / loathe elizabeth / relations / flatter elizabeth / relations / unflatter elizabeth / relations / cause chaos / relations / relieve chaos / relations".

Test me with "test acquaintance / test impression / test notice / test love / test alliance / test fancying / test marriage".

---

232 **Example** **Swerve left? Swerve right? Or think about it and die?** *RB*

Building a marble chute track in which a dropped marble will automatically roll downhill.

Suppose we have marbles that roll downhill across our map, in a life-size version of one of those marble-chute toys. We might now want to keep track of both compass relationships and which-room-slopes-into-which, so we make a new relation:

"Swerve left? Swerve right? Or think about it and die?"

```
Overlooking relates various rooms to various rooms.


The verb to overlook means the overlooking relation.


A thing can be spherical or lumpy. A marble is a kind of
thing. A marble is always spherical. The player carries a
marble called a red marble. The player carries a marble
called an agate marble. The player carries a marble
called a blue cloudy marble.


The Long Yellow Slide is north of the Funnel. The Long
Yellow Slide overlooks the Blue Funnel. The Ski-jump is
below the Blue Funnel. The Blue Funnel overlooks the
Ski-jump. The Ski-jump overlooks the Landing Bowl.
The Landing Bowl overlooks the Snake Run. The
Landing Bowl is north of the Snake Run. The Snake
Run overlooks the Goal. The Snake Run is north of the
Goal.


Definition: a room is sloping if it overlooks a room.
```

And let's say we want the player to be allowed to slide, too, since that would be much more fun than just watching the marbles go:

```
Understand "sit" as sitting down. Sitting down is an
action applying to nothing. Check sitting down: if the
player is spherical, say "You are already seated." Carry
out sitting down: now the player is spherical. Report
sitting down: say "You sit, ready to slide wherever fate
takes you."


Understand the command "stand" as something new.
```

```
Understand "stand" or "stand up" as standing up.
Standing up is an action applying to nothing. Check
standing up: if the player is lumpy, say "You are already
standing." Carry out standing up: now the player is
lumpy. Report standing up: say "You get to your feet."
```

Now a rule to control what happens to all our sliding and rolling objects:

```
Every turn:
    repeat with item running through spherical things
which are in sloping rooms:
        let the current space be the holder of the item;
        let the final space be a random room which is
overlooked by the current space;
        if the player can see the item and the item is a
marble, say "[The item] rolls out of the room toward [the
final space].[line break]";
        if the player is the item, say "You keep sliding...";
        move the item to the final space;
        if the player can see the item and the item is a
marble, say "[The item] rolls into the room from [the
current space].[line break]".
```

Since the Ski-jump overlooks the Landing Bowl, the marble will be able to fly through the air to its destination, even though there is no map connection to allow the player to cross. We might want to let the player make it across this barrier also, so:

```
Instead of jumping in a sloping room:
    say "You leap...";
    move the player to a random room overlooked by the
location.
```

Because overlooking is various-to-various, we could include that element popular in marble chute toys, the splitter:

```
The Downhill Splitter is north of the Long Yellow Slide.
"The green plastic chute runs downhill towards a Y-
junction, forcing incoming marbles right or left."
```

```
The Downhill Splitter overlooks the Long Yellow Slide
```

and the Purple Snaking Passage. The Purple Snaking Passage is southeast of the Downhill Splitter. The Purple Snaking Passage overlooks the Landing Bowl. The Purple Snaking Passage is above the Landing Bowl.

The player is in the Downhill Splitter.

Test me with "drop red / drop blue / sit / z / stand up / drop agate / sit / z / z / z / z / z".

---

233 **Example** **Beneath the Surface** *RB*

An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

The standard world model provides for the idea of containers and supporters, but this is not the only way that objects can relate to one another in the real world. Here we try adding the idea of concealment beneath another object:

"Beneath the Surface"

Section 1 - In Which our Terms are Defined

Underlying relates various things to one thing. The verb to underlie means the underlying relation. The verb to be under means the underlying relation. The verb to be beneath means the underlying relation.

Instead of looking under a thing which is underlaid by something (called the lost object):

say "You find [the list of things which underlie the noun]!";
now every thing which underlies the noun is carried by the player;
now every thing which underlies the noun does not underlie the noun.

Hiding it under is an action applying to one carried thing and one thing. Understand "put [something preferably held] under [something]" as hiding it under. Understand "hide [something preferably held] under [something]" as hiding it under. Understand the commands "shove" and "conceal" and "stick" as "hide".

Check hiding it under:
if the second noun is not fixed in place, say "[The second noun] wouldn't be a very effective place of concealment." instead.

Carry out hiding it under:
now the noun is nowhere;
now the noun underlies the second noun.

Report hiding it under:
say "You shove [the noun] out of sight beneath [the second noun]."

Section 2 - In Which They are Put To Use

The Room of Hidden Objects is a room. It contains a sofa, an easy chair, and a rug. The sofa supports a lime-green pillow and an innocent-looking Chinese finger toy. The rug is fixed in place. The chair is a supporter.

A treasure map underlies the easy chair. A skeleton is beneath the sofa. A blueprint of Atlantis, a lexicon of Linear A, and the key to Jimmy Hoffa's Mausoleum are under the rug.

Test me with "look under the sofa / look under the rug / look under the easy chair / hide lexicon under rug".

Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

We have two things to keep track of with our layering clothing: what currently is covering something else; and what *can* cover something else. This implementation goes for a fairly simple treatment, assuming that each item of clothing will completely conceal those beneath it, and that we are not implementing entire sets of shirts, jackets, etc. But it will do for a demonstration.

"Bogart"

Section 1 - Clothing Behavior

First we make our relation to represent what *is* underneath another item:

Underlying relates one thing to various things. The verb to underlie means the underlying relation. The verb to be under implies the underlying relation.

And now we prevent taking a lower layer off before the thing that is worn over it:

Before taking off something which underlies something

```
(called the impediment) which is worn by the player:
say "(first removing [the impediment])[command
clarification break]";
silently try taking off the impediment;
if the noun underlies something which is worn by the
player, stop the action.

Check taking off:
    if the noun underlies something (called the
impediment) which is worn by the player, say "[The
impediment] [are] in the way." instead.

Carry out taking off:
    now the noun is not underlaid by anything.

Report taking off something:
    say "[We] [are] now wearing [a list of uppermost
things worn by the player]." instead.

Definition: a thing is uppermost if it is not under
something.
```

That covers order of clothing removal, but we also want to restrict what can be worn on top of what else. This time we need Inform to have some idea of what customarily can be layered on top of what other clothing:

```
Overlying relates one thing to various things. The verb
to overlie means the overlying relation.

Covering relates a thing (called A) to a thing (called B)
when the number of steps via the overlying relation from
A to B is greater than 0. The verb to cover means the
covering relation.
```

With these definitions, we can say that a jacket should go over a shirt and a shirt over an undershirt (say), and then Inform will know that a

jacket will cover both shirt and undershirt.

```
Before wearing something when something (called the
impediment) which covers the noun is worn by the
player:
    while the player wears something which covers the
noun:
        say "(first removing [the impediment])[command
clarification break]";
        silently try taking off the impediment;
        if the player is wearing the impediment, stop the
action.



Carry out wearing:
    if the noun covers something (called the hidden item)
worn by the player, now the hidden item underlies the
noun.



Instead of looking under something which is worn by the
player:
    if something (called the underwear) underlies the
noun, say "[We] [peek] at [the underwear]. Yup, still
there.";
    otherwise say "Just [us] in there."



Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying
[a list of things carried by the player][else][We]['re]
empty-handed[end if][if the player wears something].
[We] [are] wearing [a list of uppermost things worn by
the player][end if]."



To peek is a verb.
```

Notice that our inventory only describes the things that the player can see as the upper layer of clothing.

```
Section 2 - The Scenario

The Trailer is a room. "A full-length mirror is the main
amenity in here, and that suits you just fine." The full-
length mirror is scenery in the Trailer. Instead of
examining or searching the mirror, try taking inventory.

The player wears a fedora, a jacket, a shirt, some
undershorts, an undershirt, some slacks, a pair of
socks, and a pair of shoes.

The shirt underlies the jacket. The pair of socks
underlies the pair of shoes. The undershorts underlie
the slacks. The undershirt underlies the shirt.

The jacket overlies the shirt. The shoes overlie the
socks. The slacks overlie the undershorts. The shirt
overlies the undershirt.

Test me with "x mirror / remove fedora / remove jacket /
remove shirt / remove slacks / remove undershirt /
remove shoes / remove socks / remove shorts / remove
undershorts".
```

If we further wanted to prevent the player from taking off clothes in inappropriate places, we might add something like this:

```
Instead of taking off something in the presence of
someone who is not the player:
    say "[We] [are] far too modest to strip in public."
```

Example **The Problem of Edith**

A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before. *RB*

Suppose that we have a core set of issues we want to be able to bring up with all the characters, and we want characters to draw intelligent connections between different conversation topics. We will need some model of how things relate to one another, so:

```
"The Problem of Edith"

Suggestion relates things to each other. The verb to
suggest means the suggestion relation.

A subject is a kind of thing. The current subject is a
thing that varies. greeting is a subject.

Understand "ask [someone] about [any subject]" as
asking it about the subject. Understand "tell [someone]
about [any subject]" as asking it about the subject.

Asking it about the subject is an action applying to one
thing and one visible thing.

Carry out asking it about the subject:
    say "'Hmm, [the second noun],' says [the noun]. ";
    relate the current subject with the second noun;
    now the current subject is the second noun.
```

And if we wanted to offer the player some hints about angles he could pursue:

```
Instead of thinking:
```

```
    say "You contemplate [a list of things suggested by
the current subject]."
```

For that matter, we could use the same system to have characters make sense of any physical evidence the character shows them:

```
Instead of showing something which suggests the
current subject to someone:
    say "[The second noun] nods impatiently."


Instead of showing something to someone:
    let the next subject be the next step via the
suggestion relation from the noun to the current subject;
    if the next subject is a subject:
        try asking the second noun about the subject the
next subject;
    otherwise:
        say "[The second noun] shrugs."


When play begins:
    now the left hand status line is "Discussing: [current
subject]";
    now the right hand status line is " ".


Broughton Hall is a room. Lady Uckfield is a woman in
Broughton Hall. "Lady Uckfield sits at her desk, looking
wholly composed."


The nasty letter is a thing carried by the player. The
nasty letter suggests infidelity and penmanship. The
ten-pound note is carried by the player. It suggests
money.


Infidelity is a subject. Infidelity suggests marriage and
divorce. Marriage suggests love. Marriage, love, and
divorce are subjects.
```

```
Penmanship is a subject. Penmanship suggests
education. Education is a subject. Class status and
money are subjects. Class status suggests education.
Money suggests class status and marriage.
```

```
The current subject is divorce.
```

Now we can define what gets said when the subject is changed, regardless of whether the segue was introduced in speech or by a shown object. Since rows are blanked after use, the speaker will never repeat herself; if we provide more than one line about the same pair of topics, the first one will be used, then the second, and so on, until the table runs out:

```
To relate (initial - a subject) with (next - a subject):
    repeat through Table of Remarks:
        if the initial is starting entry and the next is the final
entry:
            say "[comment entry][paragraph break]";
            blank out the whole row;
            rule succeeds;
    say paragraph break.
```

```
Table of Remarks

initial                 next                    comment
starting                divorce                 "'As it seems to me, all the love is on one side,' she says crisply. 'And that rarely works.'"
divorce                 penmanship              "'Stop making that plea: it won't work.'"
infidelity              divorce                 "'Frankly, I rather think there would have been cause enough for divorce without the perversely plentiful evidence of unfaithfulness.'"
divorce                 money                   "'I mean that the divorce will be expensive, I know it,' she says. 'But I can think of no happier investment.'"
marriage                infidelity              "'If you wish me to understand that it was a marriage for money, you could have spared your energy. That was patent from the outset.'"
infidelity              penmanship              "'Infidelity, but I don't see how having married for money excuses a subsequent infidelity.'"
```

If we had more than one character in the scenario, we could provide multiple tables, but this will do to demonstrate the idea.

Of course, we can override specific instances, if we want the character always to say the same thing regardless of how we came to this point:

```
Instead of asking Lady Uckfield about the subject
penmanship:
    now the current subject is penmanship;
    say "She sighs. 'So few people write really beautifully
these days.'"
```

```
Test me with "think / ask lady about infidelity / show
nasty letter to lady / show note to lady / think / ask lady
about divorce / ask lady about love / ask lady about
marriage / ask lady about divorce / ask lady about love /
ask lady about penmanship".
```

We would have to be careful about this system, since we have applied a various-to-various relation to every single object in the game. In practice it would probably be wisest to restrict it a bit, with judicious definitions of kind and so on.

Example: Wainwright Acts

RB

A technical note about checking the location of door objects when characters other than the player are interacting with them.

Suppose we wanted to write rules for a character who will interact with doors in other locations even when the player is not present. This poses a little challenge: doors are actually single objects, and -- with the same shuffling of stage properties that applies to backdrops -- they are moved as needed to represent the door object in whatever room contains the player.

That means that it isn't safe to rely on a phrase like

```
if an open door is in the location of Bernard
```

because, even if Bernard's location is connected by doors to other places, the actual representation of that door may not be "in" Bernard's location, from the model's point of view, at this exact moment.

This does not, of course, mean that we can't ask this question; just that we have to be a little cleverer about how we phrase it. Every door has properties that correspond to the two locations

linked:

```
the front side of the blue door (a room, which is
arbitrarily one side of the door)
the back side of the blue door (arbitrarily the other side)
```

We can make this information easier to check with a conditional relation, like so:

```
Liminality relates a door (called X) to a room (called Y)
when the front side of X is Y or the back side of X is Y.
The verb to be a threshold of means the liminality
relation.
```

And this allows us to write rules that have characters interacting with doors even in the player's absence:

```
"Wainwright Acts"


The Waiting Room is a room. The waiting room door is
west of the Waiting Room and east of the Gents' Loo.
The Waiting Room door is an open door. "The waiting
room door [if open]stands open[otherwise]is shut
firmly[end if]."


Sir Humphrey is a man in the Gents' Loo.


Liminality relates a door (called X) to a room (called Y)
when the front side of X is Y or the back side of X is Y.
```

```
The verb to be a threshold of means the liminality
relation.


Definition: a person is other if he is not the player.


Every turn:
    repeat with indiscreet one running through other
people:
        repeat with port running through open doors that
are a threshold of the location of the indiscreet one:
            if the port is a threshold of the location and the
indiscreet one is not in the location:
                say "Through [the port], you overhear [the
indiscreet one] discussing [one of]his hopes for your
imminent resignation[or]your wife's infidelity[or]your
financially straitened circumstances[or]ways to avoid
attending your birthday party[or]your halitosis[as
decreasingly likely outcomes]."


Test me with "z / z / z / w / z / e / close door / z".
```

---

237 **Example** **A Humble Wayside Flower** *RB*

Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

First we define the relationships we choose to acknowledge:

```
"A Humble Wayside Flower"


Marriage relates one person to another (called the
spouse). The verb to be married to means the marriage
relation.
```

```
Fatherhood relates one person (called father) to various
people. The verb to engender means the fatherhood
relation.
```

For brevity, we will ignore the existence of mothers. It is a sad world.

```
Siblinghood relates a person (called A) to a person
(called B) when a person who engenders A engenders
B. The verb to be sibling to means the siblinghood
relation.
```

```
Family relates a person (called A) to a person (called B)
when A is married to B or A engenders B or B
engenders A or A is sibling to B. The verb to be related
to means the family relation.
```

```
A person can be known or unknown. After printing the
name of an unknown person (called the alien):
    if a known person (called the contact) is related to the
alien:
        say " ([relation between alien and contact] of [the
contact])";
        now the alien is known;
        rule succeeds.
```

```
To say relation between (first party - a person) and
(second party - a person):
    if the first party is married to the second party:
        if the first party is female, say "wife";
        otherwise say "husband";
        rule succeeds;
    if the first party is sibling to the second party:
        if the first party is female, say "sister";
        otherwise say "brother";
        rule succeeds;
    if the first party engenders the second party:
        say "father";
        rule succeeds;
    if the second party is the father of the first party:
        if the first party is female, say "daughter";
        otherwise say "son";
```

rule succeeds.

```
Pere Blanchard's Hut is a room. Percival Blakeney is a
known man in the Hut. Marguerite is a woman in the
Hut. Percival is married to Marguerite. Outside from the
Hut is the Garden. Louise is a woman in the Garden.
The Road to Paris is west of the Garden. Armand St
Just is a man in the Road. Louise is married to Armand.
Monsieur St Just is a man. He engenders Armand and
Marguerite.
```

```
Test me with "out / west / east / west".
```

Monsieur St Just never appears on the scene in this piece, but if we did put him somewhere the player could find him, he, too, would be properly introduced.

---

238 **Example** **Meet Market** *RB*

A case in which relations give characters multiple values of the same kind.

We have already seen that we can give things value properties -- a lamp has a brightness, for instance. Relations give us additional flexibility: since we may relate various things to various values, it is possible to describe a thing as having more than one value at the same time.

To demonstrate:

```
"Meet Market" by "K M and Eric Rossing"
```

```
Feature is a kind of value. The features are snub-nosed,
gangly, comely, bright-eyed, and sulky.
```

```
Appearance relates various persons to various features.
```

The verb to appear means the appearance relation.

Meet Market is a room.

Bob, Carol, Ted, and Alice are people in the Meet Market.

Bob appears snub-nosed and gangly.

Ted appears sulky and snub-nosed.

Carol appears comely and bright-eyed.

Alice appears bright-eyed and comely.

Yourself appears sulky and gangly.

Instead of looking:
    say "The snub-nosed ones: [list of people who appear snub-nosed][line break]";
    say "The gangly ones: [list of people who appear gangly][line break]";
    say "The comely ones: [list of people who appear comely][line break]";
    say "The bright-eyed ones: [list of people who appear bright-eyed][line break]";
    say "The sulky ones: [list of people who appear sulky][paragraph break]".

```
Test me with "look".
```

The same logic might be used to provide characters who have complex mood states: a person might be angry and sad, not merely one or the other -- feelings being what they are.

---

239 **Example** **For Demonstration Purposes** *RB*

A character who learns new actions by watching the player performing them.

Suppose we want to have a character who can dynamically learn new actions by observing the player performing them. We could do this by adding the actions to a list of things the character can do, but using a relation to express the same idea allows for tidier, easier-to-read code.

Thanks to Jesse McGrew for the initial design of this example.

```
"For Demonstration Purposes"


Section 1 - Procedure


Capability relates various people to various stored
actions. The verb to be capable of means the capability
relation.


Persuasion rule:
    let CA be the current action with no specific actor;
    if the person asked is capable of CA:
        persuasion succeeds;
    otherwise:
        say "[The person asked] look[s] confused. Maybe a
demonstration would help.";
        persuasion fails.
```

The action requester is an object that varies. The action requester variable translates into I6 as "act_requester".

To decide which action is the current action with no specific actor:
    let old actor be the person asked;
    let old requester be the action requester;
    now the person asked is the player;
    now the action requester is nothing;
    let CA be the current action;
    now the person asked is the old actor;
    now the action requester is the old requester;
    decide on CA.

The learning by observation rule is listed after the report stage rule in the specific action-processing rules.

Definition: a person is other if he is not the player.

This is the learning by observation rule:
    repeat with the viewer running through other people who can see the player:
        if the player is the actor and viewer is not capable of the current action:
            say "[The viewer] watches your behavior with interest. Seems like [they] [are] learning.";
            now the viewer is capable of the current action.

Section 2 - Scenario

The Daily Planet is a room. Clark is here. He is a man.

When play begins:
    now Clark is capable of taking inventory.

Test me with "Clark, inventory / Clark, x me / x me / Clark, x me".

---

240 **Example** **Number Study** *RB*

The parity and joint magnitude relations explored.

This is the same case given above, but expanded just slightly to demonstrate that the names of the relations can also be printed, if we like:

```
"Number Study"

Abstraction is a room.

Parity relates a number (called N) to a number (called M) when N minus M is even.

Joint magnitude relates a number (called N) to a number (called M) when N plus M is greater than 7.

To chart (R - a relation of numbers):
    repeat with N running from 1 to 5:
        repeat with M running from 1 to 5:
            if R relates N to M, say "[N] <=> [M] by [R][line break]";

When play begins:
    let L be { parity relation, joint magnitude relation };
```

```
repeat with R running through L:
    chart R.
```

As this shows, we can even form lists of relations. The kind of L is "list of relations of numbers".

---

241 Example Murder on the Orient Express *RB*

A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but there is physical evidence which may change their minds...

The following example creates two new relations, and two new verbs, in order to set up a tangled web of intrigue.

```
"Murder on the Orient Express"

The Dining Car is a room. Lord Peter is a man in the
Dining Car. Sherlock Holmes is a man in the Dining Car.
Miss Marple is a woman in the Dining Car. Adam
Dalgliesh is a man in the Dining Car.

Suspecting relates various people to one person.

The verb to suspect means the suspecting relation.

Dalgliesh suspects Holmes. Holmes suspects Lord
Peter. Lord Peter suspects Holmes. Miss Marple
suspects the player.

Exculpating relates one thing to various people.
```

```
The verb to exculpate means the exculpating relation.
```

```
The silver bullet exculpates the player. The pipe ash
exculpates Holmes. The poison pen letter exculpates
Lord Peter. The poison pen letter exculpates Miss
Marple. [Poor Dalgliesh. I guess he did it.]
```

```
The pipe ash, the letter and the silver bullet are carried.
```

Given this, we can then set up elaborate rules:

```
Instead of showing something to a person who suspects
the player:
    say "'You would say that,' remarks [the second noun]
darkly.".
```

```
Instead of showing something which exculpates the
player to someone:
    say "'How striking!' says [the second noun]. 'Almost I
begin to distrust myself.'".
```

```
Test me with "show the letter to miss marple / show the
silver bullet to holmes".
```

And so on: "if Dalgliesh suspects someone who is exculpated by something carried by the player...", for instance, makes a fitting final example for this chapter. The description

```
someone who is exculpated by something carried by the
player
```

expresses a complicated idea in very few words, and in such a way that a passer-by looking at the source text would immediately see what was meant.

The moral is that relations allow sophisticated patterns of behaviour to be created in a way that reads back naturally as English.

 Example **What Not To Wear** *RB*

A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

"What Not To Wear"

Section 1 - Overlying and Underlying

We start by borrowing some of the same ideas from the Bogart example, but we're also going to make a kind called "garment-element". This kind will include both garments (objects of clothing) and body parts (things that can be covered by clothing); using it allows us to restrict the way our underlying and overlying relations apply, which will make them a bit faster at run-time.

```
A garment-element is a kind of thing.

Underlying relates various garment-elements to various
garment-elements with fast route-finding. The verb to
underlie means the underlying relation. The verb to be
under implies the underlying relation.

Check taking off:
    if the noun underlies something (called the
impediment) which is worn by the player, say "[The
impediment] [are] in the way." instead.

Carry out taking off:
```

```
    now the noun is not underlaid by anything.

Report taking off something:
    say "[We] [are] now wearing [a list of uppermost
things worn by the player]." instead.

Definition: a garment-element is uppermost if it is not
under something opaque.
```

Here we've expanded on the previous ideas of 'uppermost' because it is possible for an upper layer to reveal what lies beneath: a tie, a clear plastic trenchcoat, an open-knit sweater, etc. We'll make such items transparent.

```
Before taking off something which underlies something
which is worn by the player:
    while the noun underlies something (called the
impediment) which is worn by the player:
        say "(first removing [the impediment])[command
clarification break]";
        silently try taking off the impediment;
        if the noun underlies the impediment, stop the
action.

Overlying relates various garment-elements to various
garment-elements. The verb to overlie means the
overlying relation.

Covering relates a garment-element (called A) to a
garment-element (called B) when the number of steps
via the overlying relation from A to B is greater than 0.
The verb to cover means the covering relation.

Before wearing something when a garment which
covers the noun is worn by the player:
    while the player wears a garment (called the
```

```
impediment) which covers the noun:
        say "(first removing [the impediment])[command
clarification break]";
        silently try taking off the impediment;
        if the player is wearing the impediment, stop the
action.


Carry out wearing:
    repeat with hidden item running through things worn
by the player:
        if the noun covers the hidden item, now the hidden
item underlies the noun.


Instead of looking under something which is worn by the
player:
    if something (called the underwear) underlies the
noun, say "[We] [peek] at [the underwear]. Yup, still
there.";
    otherwise say "Just [us] in there."


Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying
[a list of things carried by the player][else][We]['re]
empty-handed[end if][if the player wears something].
[We] [are] wearing [a list of uppermost garments worn
by the player][end if]."


To peek is a verb.
```

Section 2 - Regional Coverage

Here we draw in the idea that different clothes go over different areas of the body, and that they should be in competition with each other only if both sets of clothes belong at the same level over the same body area.

```
Before wearing something:
```

```
    let N be the layering depth of the noun;
    repeat with item running through things worn by the
player:
        if the layering depth of the item is N and the item
covers a body-part which is covered by the noun:
            say "(first taking off [the item])[command
clarification break]";
            silently try taking off the item;
            if the player wears the item, stop the action.
```

This may seem like overkill, but it allows us to create garments that cover different subsets of the body -- pants and shirt vs. a dress, for instance.

```
To decide what number is the layering depth of (chosen
garment - a thing):
    let N be 0;
    if the chosen garment covers a body-part (called
base):
        let N be the number of steps via the overlying
relation from the chosen garment to the base;
    decide on N.
```

To help with modeling, we'll give everyone body parts, broken down according to their relevance to clothing:

```
A body-part is a kind of garment-element. A torso, a
seat, a head, pair of legs, and pair of feet are kinds of
body-part.
```

If we wanted to allow gloves, we might put in hands as well; but this is enough for now.

```
One head is part of every person. One torso is part of
every person. One pair of legs is part of every person.
One pair of feet is part of every person. One seat is part
of every person.
```

And now we make some categories of clothing:

```
A garment is a kind of garment-element. A garment can
be transparent. A pair of pants, a pair of underpants, a
foundation garment, a pair of socks, a pair of shoes, a
```

jacket, a hat, a dress, and a shirt are kinds of garment.

The plural of pair of pants is pairs of pants. The plural of pair of underpants is pairs of underpants. The plural of pair of socks is pairs of socks. The plural of pair of shoes is pairs of shoes.

A pair of pants, a pair of underpants, a foundation garment, a pair of socks, a pair of shoes, a jacket, a hat, a dress, and a shirt are usually wearable.

When play begins:
    now every pair of socks overlies every pair of feet;
    now every pair of shoes overlies every pair of socks;
    now every pair of underpants overlies every seat;
    now every pair of pants overlies every pair of underpants;
    now every foundation garment overlies every torso;
    now every jacket overlies every shirt;
    now every jacket overlies every dress;
    now every hat overlies every head;
    now every dress overlies every pair of underpants;
    now every dress overlies every foundation garment.

Section 2 - The Scenario

The Dressing Room is a room.

The player carries some capris, some jeans, a corset, a plunge bra, a thong, boy-shorts, black satin D'Orsay pumps, brown leather boots, a camisole, a cocktail dress, a bolero, a cashmere shrug, a sheer wrap, and a linen tunic.

```
The woolly socks are a pair of socks.
The D'Orsay pumps and the brown leather boots are
pairs of shoes.
The thong and the boy-shorts are pairs of underpants.
The capris and the jeans are pairs of pants.
The tunic is a shirt.
The camisole, the corset, and the plunge bra are
foundation garments.
The cocktail dress is a dress.
The bolero, the cashmere shrug, and the sheer wrap
are jackets. The shrug and the wrap are transparent.
```

```
Test me with "wear capris / wear jeans / i / wear thong /
i / wear dress / wear corset / wear dress / i / wear wrap /
i / wear boots / wear pumps / i".
```

---

243 **Example** **Mathematical view of relations** *RB*

Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

Inform uses the term "relation" in a broader sense than mathematics. Properly speaking, the term "relation" in its mathematical sense only applies to the case where the domain for the left and right objects are the same: for simplicity's sake, let us talk only about the case where they are.

In mathematics, the properties most often looked for in a relation are that it should be:

(a) Reflexive: A <=> A for every A. This is not especially useful for Inform, and seldom appears in practical examples.

(b) Symmetric: A <=> B if and only if B <=> A. Generally, Inform relations are not symmetric, but there are two important cases which are:

```
Meeting relates people to each other.
Marriage relates one person to another.
```

These are automatically symmetric, so that to assert one way round is to assert the other as well.

(c) Transitive: A <=> B and B <=> C means that A <=> C as well. Again,

Inform relations are not generally transitive. In many relations, there can be long chains of things, each perhaps related to the one in front and the one behind, so that there is some indirect sense in which the two ends of the chain are connected to each other: but they are not related as such. For instance, a journey across the map might pass through ten rooms, each adjacent to the last and next, but the two ends would not themselves be adjacent. The concept we need is the "transitive closure" of the original relation, defined as the smallest transitive relation including the original. If R is a relation between "things", then the following:

```
TC relates a thing (called A) to a thing (called B) when
the number of steps via R from A to B is greater than 0.
```

is the transitive closure of R. In particular,

```
Accessibility relates a room (called A) to a room (called
B) when the number of moves from B to A is greater
than 0. The verb to be accessible from means the
accessibility relation.
```

calculates the transitive closure of adjacency. Here, though, the way we normally understand "accessible from" suggests that it would be better to write:

```
Accessibility relates a room (called A) to a room (called
B) when the number of moves from B to A is at least 0.
```

which is reflexive as well as transitive. The usefulness of Inform's "next step via R from A to B" construction, in a wide variety of settings, reflects the importance of transitivity as an idea.

A relation which has all three properties of being reflexive, symmetric and transitive is called an "equivalence relation". (If all the map connections are two-way, then the accessibility relation above is symmetric and therefore a full equivalence relation: but if not, it may not be.) Inform has a special construction for making equivalence relations:

```
Nationality relates people to each other in groups.
```

This language - "in groups" - relies on the standard theorem that every equivalence relation on a set naturally defines a partition of that set, and

vice versa. The "groups" referred to are what are normally called "equivalence classes". (Inform does little with these equivalence classes: it might be interesting to do so, in effect forming quotient kinds.)

 Example **Graph-theory view of relations** *RB*

Some notes on relations from the point of view of graph theory.

One way to look at a relation is to regard it as a directed graph: that is, a collection of things ("vertices") with arrows drawn between them ("edges"). We write our items A, B, C, ... on a piece of paper: then, if A relates to B, we draw an arrow pointing from A to B, and so on. If we made this drawing for the adjacency relation, we would more or less have reconstructed the map (or at least a simplified one which does not care about precise directions, like the famous diagram of the London Underground). But the drawing can be made for any relation. If we define:

```
Suspecting relates various people to one person.
```

then, in the corresponding graph, each "vertex" will have at most one arrow leading away from it - though there could be many (or none) leading towards. Conversely, a one-to-various relation produces a graph where each vertex has at most one arrow coming in. A one-to-one relation means that the picture consists of some vertices on their own, with no arrows, a few perhaps with looped arrows leading from and to themselves, and then a collection of pairs joined by arrows. On the other hand, a various-to-various relation is just a free-for-all, with no restrictions on the arrows. The relations:

```
Meeting relates people to each other.
Marriage relates one person to another.
```

always have the property of working both ways round, and these are easiest to visualise by forgetting the direction of the arrows, so that they just become lines joining the vertices.

Inform uses a different algorithm for finding routes ("the next step via R from A to B") in each of these cases, and internally it stores relations in different formats in the different cases, because it makes a big difference to the efficiency of Inform to minimise the storage required for a relation

and the time taken to explore it.

All the cases are benign except for "various to various" - the most useful - and for its closely related symmetrical version, "relates... to each other". Inform cannot afford to assume that the relation will be "sparse" (that is: that no vertex will have more than a certain number of arrows, or that the total number of arrows will be small), because it has no idea how the arrows will come and go during play. It therefore uses 1 bit of storage for each pair of objects. This sounds harmless, but if there are 200 rooms, there are 40,000 pairs of rooms, which means a 5000-byte allocation of storage (plus a handful of bytes as overhead). Gratuitous various-to-various relations are therefore not a good idea.

There is a standard algorithm for calculating shortest paths through a directed graph, but Inform does not always use it, because there is not always memory to store the required matrix of partial results. Inform's slow method, likely to be used on the Z-machine, requires a storage overhead which is equal to the number of vertices, not the square of that number, but the worst-case running time can be bad: if there are N vertices, and the diameter of graph (the longest distance between vertices) is D, then the running time is proportional to D times N. The worst case in finding routes from A to B is when almost every vertex can reach B, some across long trails, but A cannot. In the case of finding routes across the game's map, this must be multiplied further by the number of possible directions - usually 16.

This does not sound too awful, but if one is trying to find (say) "the most distant room from A", that means a further loop and now the running time will be D times N squared. Extension writers will need to be careful of this kind of thing: it is easy to write highly cool prototypes which work terribly slowly on larger, more realistic maps.

Chapter 14: Adaptive Text and Responses

things; *§14.6. Adapting demonstratives and possessives*; *§14.7. Can, could, may, might, must, should,*

Contents of *Writing with Inform*
Chapter 13: Relations
Chapter 15: Numbers and Equations
Indexes of the examples

§14.1. Tense and narrative viewpoint

A conspicuous difference between interactive fiction and a traditional novel is the point of view from which it's told. Inform usually produces text like:

```
You can see a grey cat in the basket.
```

where a novel would usually write:

```
He saw a grey cat in the basket.
```

Standard interactive fiction (IF) is second person singular, and present tense; most novels are told in the third person singular, and past tense.

But these are just conventions - a few novels, for example, use the so-called present historic ("Napoleon looks up at the sky and sighs. Must Ney always be so doubting?"), and plenty are told in the first person singular ("I always get the shakes before a drop."). Inform allows some of this flexibility, too. The two values:

```
story viewpoint
story tense
```

control the style of the text produced. The story viewpoint has to be one of the values:

```
first person singular
second person singular
third person singular
first person plural
second person plural
third person plural
```

(which are actually the six possible values of a kind called "narrative viewpoint"), while the story tense must be one of:

```
past tense
present tense
future tense
perfect tense
past perfect tense
```

(from a kind called "grammatical tense"). Combining these gives 30 possibilities in all, though only a few are at all commonly used.

It's important to make a very large caveat here: Inform uses these settings in producing the replies ("responses") by the built-in actions, but the only way for all of our own text to have a particular tense or narrative viewpoint is to write it that way. If we write:

```
The Taj Mahal is a room. "You stand and admire the Taj Mahal."

When play begins:
    now the story viewpoint is first person plural;
    now the story tense is past tense.
```

then we're likely to see the following peculiar transcript:

```
Taj Mahal
You stand and admire the Taj Mahal.

>e
We couldn't go that way.
```

That's because the response ("We couldn't go that way") was constructed to follow

the settings for viewpoint and tense, but the fixed text of the room description wasn't. In fact there are ways to write the room description so that it would adapt itself automatically, as we'll see, but it takes a fair amount of work. More simply:

```
The Taj Mahal is a room. "I stood and admired the Taj Mahal."

When play begins:
    now the story viewpoint is first person plural;
    now the story tense is past tense.
```

In short, tense and viewpoint switching is neat, but it isn't magic.

If we want to write text which will work in whatever the current tense is, the following turn out to be useful little conveniences:

say "[here]"

Produces "here" if the story tense is the present tense, and "there" otherwise.

say "[now]"

Produces "now" if the story tense is the present tense, and "then" otherwise.

Start of Chapter 14: Adaptive Text and Responses
Back to Chapter 13: Relations: §13.16. What are relations for?
Onward to §14.2. Adaptive text

§14.2. Adaptive text

Paying attention to the tense and viewpoint is one reason why text might need to adapt. Another is that it might need to adapt according to whether nouns are singular or plural, or whether it talks about the player or some third party. For example, the following rule isn't ideal:

```
Instead of taking: say "[The noun] is pinned down by Dr Zarkov's
force field."
```

Most of the time it's fine ("The V-ray is pinned down by Dr Zarkov's force field"), but then:

```
> GET ME
You is pinned down by Dr Zarkov's force field.
> GET CONDENSERS
The condensers is pinned down by Dr Zarkov's force field.
```

Which is a little unfortunate. But the correction is very easy:

```
Instead of taking: say "[The noun] [are] pinned down by Dr Zarkov's
force field."
```

The result is much better: "The V-ray is pinned down..."; "You are..."; "The condensers are...". In fact, it's also convenient because it adapts to the story viewpoint and story tense: "The condensers will be pinned down..."; "He was pinned down...".

How does Inform do this? The answer is not that "[are]" is a specially-written text substitution. In fact Inform can do this with any verb that it has a definition of. For example,

```
"[The noun] [carry] too much static charge."
```

would also adapt itself - "The V-ray carries too much static charge", and so on. There aren't many verbs built in to Inform, but "[have]" and "[carry]" and "[wear]" and "[can]" may be useful, and "[can see]" and "[can touch]". Negative forms like "[are not]" are also available:

```
"[The noun] [cannot touch] the ionizer terminal."
```

might produce "The V-ray will not be able to touch the ionizer terminal.", for example.

As these examples hint, the verb adapts itself to the most recently printed object name. All of this only works if the previous object's name is printed from a substitution. So:

```
"[The condensers] [are] working."
```

will work -- correctly forming "The condensers are working.", "The condensers will be working." or "The condensers were working.", according to the story tense -- but

```
"The condensers [are] working."
```

probably won't work. Inform doesn't have any way to understand the raw text outside of the text substitution marks "[" and "]", and it doesn't recognise "The condensers" as being something's name.

Something else to be careful with is the use of lists. If we write this:

```
"[The condensers] and [the V-ray] [are] smashed by Voltan's
birdmen."
```

then Inform is likely to print:

```
The condensers and the V-ray is smashed by Voltan's birdmen.
```

because it looks at the most recently named object - the V-ray, singular - to decide whether to use "is" or "are". On the other hand, Inform gets this right:

```
"[The list of things on the bench] [are] smashed by Voltan's birdmen."
```

Because Inform constructs the list itself, it's able to appreciate that the things listed are jointly the subject of the verb, and it uses that information to decide on "is" or "are". So:

```
The condensers and the V-ray are smashed by Voltan's birdmen.
The Atomic Furnace shovel is smashed by Voltan's birdmen.
```

Start of Chapter 14: Adaptive Text and Responses
Back to §14.1. Tense and

§14.3. More on adapting verbs

If we need an adaptive message with a verb which doesn't belong to Inform's built-in set, all we need do is define it. In the previous chapter we defined verbs by giving them meanings, but in fact that's optional. For example:

```
To retrofit is a verb.
```

defines a verb without telling Inform what it means. Inform will throw a Problem message if we try to write text like:

```
Flash retrofits the meteor beam.
```

because, after all, it doesn't know what "retrofit" means. But it does still know how to print it, so this works:

```
"[The actor] [retrofit] the Mecha-Mole."
```

which might come out as "Dale retrofits the Mecha-Mole", or "Barin's archers retrofitted the Mecha-Mole", and so on.

This is especially neat for writing a single response to an action which works regardless of who the actor was. For example, the Standard Rules include:

```
say "[The actor] [put] [the noun] on [the second noun]."
```

And this can make either:

```
You put the revolver on the table.
General Lee puts the revolver on the table.
```

Start of Chapter 14: Adaptive Text and Responses
Back to §14.2. Adaptive text
Onward to §14.4. Adapting text about the player

Example 245: ★★ **Fun with Participles** Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on Clark's idle activity.

Example 246: ★★ **Variety** Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

Example 247: ★★ **Variety 2** This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

Example 248: ★★★ **Narrative Register** Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

---

§14.4. Adapting text about the player

In second-person-singular IF, the player is always "you". Many messages look like so:

```
"You have twenty minutes remaining."
```

where the subject, or the object, of the sentence is "you". But what if we want to have this text adapt itself to different narrative viewpoints?

The solution is to use the following:

```
"[We]" or "[we]"
"[Us]" or "[us]"
"[Our]" or "[our]"
"[Ours]" or "[ours]"
"[Ourselves]" or "[ourselves]"
```

The capitalised and uncapitalised versions are identical except, of course, that the initial letter of the resulting text is upper case in one but not the other. As examples of these:

```
"[We] [carry] the Queen's warrant."
"The birds drop pebbles on [us]. Right on [our] heads!"
"[Ours] [are] the burdens of office, which [we] take on [ourselves]."
```

Notice that all five of these forms are differently worded, in English. That's the reason why we use the plural to write them - the traditional second person plural forms would be "you", "you", "your", "yours" and "yourself", so we wouldn't know if "[you]" was supposed to be the subject or the object of the verb. So the convention with all of these adaptive forms is that we use "we" and its variations. (That's also why the verbs are written in the plural - "[are]", not "[is]"; "[carry]", not "[carries]".)

Start of Chapter 14: Adaptive Text and Responses

Back to §14.3. More on adapting verbs

Onward to §14.5. Adapting text referring to other things

§14.5. Adapting text referring to other things

The family in the previous section - "[we]", "[us]", "[our]", "[ours]", "[ourselves]" - always referred to the player. But we also sometimes want to refer to other things without naming them. For example, how should we adapt this?

```
> EXAMINE TREE
It has no clear outline in this misty netherworld.
```

We can easily make the verb adapt - change the "has" to "[have]" - but the trick here is to make the "It" adapt to cases where what's examined is plural, or animate. What we want is:

```
Instead of examining in the Netherworld:
    say "[regarding the noun][They] [have] no clear outline in this misty
netherworld."
```

For example, this produces:

```
> EXAMINE ME
You have no clear outline in this misty netherworld.
> EXAMINE MARK
He has no clear outline in this misty netherworld.
> EXAMINE DRUMS
They have no clear outline in this misty netherworld.
```

Note that we have to say "[regarding the noun]", not just start in with "[They]", because nothing has been named so far in the sentence - so Inform doesn't know what object it refers to. "[regarding the noun]" prints nothing, and simply tells the printing part of Inform that the subject has changed.

This isn't always needed:

```
"[We] [have] a look at [the noun], but [they] [are] just too big."
```

works fine, because printing "[the noun]" changes the subject to that, and then "[they]" agrees with it automatically. The text might come out, for example, as:

```
I had a look at Peter Rabbit, but he was just too big.
You have a look at the chessmen, but they are just too big.
We have a look at ourselves, but we are just too big.
```

We have a family of five text substitutions here, matching those in the previous section:

```
"[They]" or "[they]"
"[Them]" or "[them]"
"[Their]" or "[their]"
"[Theirs]" or "[theirs]"
"[Themselves]" or "[themselves]"
```

There's also the peculiar impersonal non-object for English sentences like "It is raining" or "There are books":

```
"[It]" or "[it]"
"[There]" or "[there]"
```

These look pointless - but consider the two texts

```
"[We] [take] [the noun]. It [rain] harder."
"[We] [take] [the noun]. [It] [rain] harder."
```

The first one risks printing "We took the scissors. It rain harder.", because it makes "[rain]" agree with "scissors", which are plural. But the second text makes "[rain]" agree with "[it]". And, as a convenience:

```
"[It's]" or "[it's]"
"[There's]" or "[there's]"
```

do the obvious thing using the current story tense.

Finally, we occasionally want to agree with a number:

```
"Honestly, [dud count][regarding the dud count] of these [are]
broken."
```

§14.6. Adapting demonstratives and possessives

Consider the following message: how might we make this adaptive?

```
> MEASURE TOP SHELF
You really are not tall enough to reach that.
```

The verbal part is easy enough, but "that" needs a new feature.

```
"[We] really [are not] tall enough to reach [regarding the noun]
[those]."
```

This could then adapt to, say,

```
> MEASURE JAM TARTS
He really was not tall enough to reach those.
```

Notice that it's "[regarding the noun][those]", not just "[those]". If we wrote "[those]", Inform would make it agree with the player, who was printed earlier in the sentence by the "[We]".

Lastly, how about:

```
> PUT TEAPOT IN MOUSEHOLE
The teapot's height is just too great.
```

This time we want:

```
"[regarding the noun][Possessive] height is just too great."
```

which might adapt to, say,

```
Our height is just too great.
Alice's height will be just too great.
```

Actually, "[regarding ...]" can be used for a description of possibly many items, too. For example:

Every turn when the player carries something:

```
say "Every possession is a worry. I wonder if [regarding things
carried by the player][they] still [look] okay in your pocket?"
```

So if the player carries just a single coin, say, this automatically becomes:

```
Every possession is a worry. I wonder if it still looks okay in your
pocket?
```

but if the player carries a pair of scissors (a single plural-named item) or a coin and an iPhone, it becomes:

```
Every possession is a worry. I wonder if they still look okay in your
pocket?
```

Once again these text substitutions are available in capitalised and uncapitalised forms:

```
"[Those]" or "[those]"
"[Possessive]" or "[possessive]"
```

In fact "[Those]" and "[those]" do subtly different things, besides the capital letter, because "[Those]" expects to be the subject of the sentence and "[those]" the object, and this makes a difference if the noun in question is a person. If the noun is an odious person called Tilly then

```
"[regarding the noun][Those] is unacceptable."
"You've never liked [regarding the noun][those]."
```

would come out as "She is unacceptable" - so "[Those]" becomes "She" - but "You've never liked her" - so "[those]" becomes "her". If we need these in different cases, we can explicitly ask for that:

```
"[those in the nominative]"
"[Those in the accusative]"
```

Start of Chapter 14: Adaptive Text and Responses
Back to §14.5. Adapting text referring to other things
Onward to §14.7. Can, could, may, might, must, should, would

Example 249: ★ **Olfactory Settings** Some adaptive text for smelling the flowers, or indeed, anything else.

§14.7. Can, could, may, might, must, should, would

English uses so-called "modal verbs" to change a sentence so that it talks about something only possibly happening. For example, the sentence "Fred goes to school" can be modified to "Fred must go to school", "Fred should go to school" or even "Fred might go to school".

Inform supports the use of modal verbs in text substitutions. For example,

```
"[Fred] [might go] to school."
```

would in the present tense come out as "Fred might go to school.", but could alternatively be "Fred might have gone to school." As this example shows, all that's needed is to take a verb we'll call V - this case, "go" - and we can write any of these:

```
"[can V]" or "[cannot V]" or "[can't V]"
"[could V]" or "[could not V]" or "[couldn't V]"
"[may V]" or "[may not V]" or "[mayn't V]"
"[might V]" or "[might not V]" or "[mightn't V]"
"[must V]" or "[must not V]" or "[mustn't V]"
```

```
"[should V]" or "[should not V]" or "[shouldn't V]"
"[would V]" or "[would not V]" or "[wouldn't V]"
```

That helps us to handle informal usages like this one:

```
"You can't go that way."
```

To make this message adaptive, we write:

```
"[We] [can't go] that way."
```

which can adapt in surprising ways -- "They won't be able to go that way.", for example.

Note that the verb V has to be one that Inform knows. But that's easy:

```
To discombobulate is a verb.
```

and then

```
"[Fred] [might not discombobulate] so easily."
```

could produce "Fred might not have discombobulated so easily", for example.

Start of Chapter 14: Adaptive Text and Responses
Back to §14.6. Adapting demonstratives and possessives
Onward to §14.8. Adapting contractions

---

§14.8. Adapting contractions

Contractions usually take the form of part of a word being missed out and replaced by an apostrophe. We've already seen "[can't]", "[couldn't]", "[mayn't]", "[mightn't]", "[mustn't]", "[shouldn't]" and "[wouldn't]", for example. But Inform supports other contractions, too, as follows.

The English verbs "to be" and "to have" are unique in having contracted forms, which we can write "['re]" and "['ve]", like this:

```
"[We]['ve] got rhythm. [We]['re] cool."
```

which might produce, say, "I've got rhythm. I'm cool.", or "He'll have rhythm. He'll be cool.", or "You had got rhythm. You were cool." (The contractions don't appear in the past tense; but the spacing fixes itself automatically.)

The Standard Rules often use a special text substitution for responses like this one:

```
"[They're] hardly portable."
```

This is exactly like "[Those]['re] hardly portable" except that if the plural is needed, Inform prints "They're hardly portable" rather than the correct, but not quite idiomatic, "Those're hardly portable". (If we wrote "[They]['re] ...", that would get the plural form right, but then the singular would be "It's hardly portable" not "That's hardly portable".)

Only a few English verbs have contracted negative forms, beyond those already mentioned. Inform knows these informal forms:

```
"[aren't]"
"[don't]"
"[haven't]"
"[won't]"
```

For example,

```
Instead of taking something:
    say "[The noun] [are] pinned down by Dr Zarkov's force field.
[They] [aren't] free to move. [They] [can't] move. [They] [won't] move.
[They] [haven't] a chance to move. Anyhow, [they] [don't] move."
```

can produce variations like these:

```
The condensers are pinned down by Dr Zarkov's force field. They
aren't free to move. They can't move. They won't move. They haven't
a chance to move. Anyhow, they don't move.
```

```
You were pinned down by Dr Zarkov's force field. You weren't free to
move. You couldn't move. You wouldn't move. You hadn't a chance
to move. Anyhow, you didn't move.
```

Start of Chapter 14: Adaptive Text and Responses
Back to §14.7. Can, could, may, might, must, should, would
Onward to §14.9. Verbs as values

§14.9. Verbs as values

Each verb known to Inform is actually a value of the kind "verb". To refer to a verb as a value, we have to put the word "verb" in front, as in these examples:

```
the verb contain, the verb might, the verb provoke
```

all of which appear in the Standard Rules.

Two adjectives are provided for use with verbs: "modal" (or "non-modal") to pick out verbs like might, could, should, and so on; and "meaningful" (or "meaningless") to pick out verbs which have a defined meaning as an Inform relation. For example, in the Standard Rules, the verb contain is meaningful, the verb might is modal, and the verb provoke is meaningless.

If V has a meaning as a relation of objects, then "meaning of V" produces that relation. For example,

```
showme the meaning of the verb contain;
showme the meaning of the verb provoke;
```

produces:

```
"meaning of the verb contain" = relation of objects: containment
relation
"meaning of the verb provoke" = relation of objects: equality relation
```

As this demonstrates, if a verb has no meaning, or its meaning doesn't relate to objects, we get just the equality relation.

In fact, Inform even defines a verb "to mean": it's meaningful, and its meaning is the meaning relation. Thus:

```
if the verb mean means the meaning relation...
```

is true. More usefully, we can search our vocabulary like this:

```
the list of verbs meaning the containment relation
```

which, unless any non-Standard Rules definitions have been added, produces:

```
list of verbs: {verb contain}
```

Note that the meaning relation can't be changed at run-time: it is not clear what it would even mean to do something like -

```
now the verb contain means the wearing relation;
```

with the story already started, so this will produce a problem message.

say "[adapt (verb)**]"**

Adapts the given verb to the current story tense and story viewpoint. For example, "you [adapt the verb provoke]" might produce "you provoke".

say "[adapt (verb) **from** (narrative viewpoint)**]"**

Adapts the given verb to the current story tense but the given viewpoint. For example, "he [adapt the verb provoke from the third person singular]" might produce "he provokes".

say "[adapt (verb) **in** (grammatical tense)**]"**

Adapts the given verb to the given tense but the current story viewpoint. For example, "you [adapt the verb provoke in the past tense]" might produce "you provoked".

say "[adapt (verb) **in** (grammatical tense) **from** (narrative viewpoint)**]"**

Adapts the given verb to the given tense and viewpoint. For example, "we [adapt the verb provoke in the future tense from the first person plural]" might produce "we will provoke".

say "[negate (verb)**]"**

Adapts the given verb to the current story tense and story viewpoint, giving it a negative sense. For example, "you [negate the verb provoke]" might produce "you do not provoke".

say "[negate (verb) **from** (narrative viewpoint)**]"**

Adapts the given verb to the current story tense but the given viewpoint, giving it a negative sense. For example, "he [negate the verb provoke from the third person singular]" might produce "he does not provoke".

say "[negate (verb) **in** (grammatical tense)**]"**

Adapts the given verb to the given tense but the current story viewpoint, giving it a negative sense. For example, "you [negate the verb provoke in the past tense]" might produce "you did not provoke".

say "[negate (verb) **in** (grammatical tense) **from** (narrative viewpoint)**]"**

Adapts the given verb to the given tense and viewpoint, giving it a negative sense. For example, "we [negate the verb provoke in the future tense from the first person plural]" might produce "we will not provoke".

Note that the verb doesn't have to be named explicitly for use by the adapt or negate phrases, so for example:

```
To decide which text is the rendering of (V - verb) (this is my
rendering):
    decide on "[negate V in the past perfect tense]".

When play begins:
    showme my rendering applied to the list of meaningful verbs.
```

produces:

```
"my rendering applied to the list of meaningful verbs" = list of texts:
{"had not had", "had not related", "had not meant", "had not
provided", "had not contained", "had not supported", "had not
incorporated", "had not enclosed", "had not carried", "had not held",
"had not worn", "had not been able to see", "had not been able to
touch", "had not concealed", "had not unlocked"}
```

Lastly, we can get at three other useful parts of a verb, too. These aren't adaptive, of course: a verb only has one infinitive form.

say "[infinitive of (verb)**]"**

Produces the infinitive of the given verb. Note that this is without a "to": for example, "[infinitive of the verb carry]" is "carry", not "to carry".

say "[past participle of (verb)**]"**

Produces the past participle of the given verb. For example, "[past participle of the verb carry]" is "carried". Warning: because modal verbs like "should" or "might" are defective in English, this will produce odd results on them - "shoulded" and "mighted", for example.

say "[present participle of (verb)**]"**

Produces the present participle of the given verb. For example, "[present participle of the verb carry]" is "carrying". Warning: because modal verbs like "should" or "might" are defective in English, this will produce odd results on them - "shoulding" and "mighting", for example.

Start of Chapter 14: Adaptive Text and Responses

Back to §14.8. Adapting contractions

Onward to §14.10. Responses

Example 250: ★★ **History Lab** We create phrases such as "the box we took" and "the newspaper Clark looked at" based on what has already happened in the story.

Example 251: ★★ **Relevant Relations** An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

---

§14.10. Responses

Most of the text which the player sees is drawn from the source, but mixed in with this are messages apparently added by Inform itself - usually in the form of short sentences saying that something has been done, or that something can't be done. Such pieces of text are called "responses", because they are almost always replies to commands. For example:

```
> EAST
You can't go that way.

> JUMP
```

```
You jump on the spot.
```

Responses like this, which don't appear anywhere in the source text, come from one of the extensions being used; most often from the Standard Rules, the "extension" which is automatically included in every project. The SR contain many small rules, and almost all of these are capable of producing one or two standard responses. These are labelled with the rule's name and then a bracketed letter - (A), (B), (C), ... as needed so that every response has its own unique name. There's nothing very mysterious about how this is done. For example, here is a rule with one response:

```
Carry out taking inventory (this is the print empty inventory rule):
    if the first thing held by the player is nothing,
        say "[We] [are] carrying nothing." (A) instead.
```

which makes the familiar text "You are carrying nothing." a response named:

```
print empty inventory rule response (A)
```

These names are actually values, belonging to the kind "response". Because of that, if we try this:

```
say "Hmm: [print empty inventory rule response (A)]"
```

Inform will produce

```
Hmm: print empty inventory rule response (A)
```

since we gave Inform a value to print, and that's just what it then did. As an alternative:

say "[text of (response)**]"**

This text substitution writes out the current text of the given response.

Thus,

```
say "Hmm: [text of print empty inventory rule response (A)]"
```

produces

```
Hmm: You are carrying nothing.
```

Start of Chapter 14: Adaptive Text and Responses
Back to §14.9. Verbs as values
Onward to §14.11. Changing the text of responses

§14.11. Changing the text of responses

These responses are named so that they can be changed. Most IF authors dislike one or two of the existing responses, and some would like to change almost all of them to give the text a different style; and extensions for IF in languages other than English change literally every response, of course.

It's very easy to change responses:

```
The print empty inventory rule response (A) is "Your hands are, like,
totally empty. Lame."
```

and we can even do this dynamically during play:

```
now the print empty inventory rule response (A) is "Your hands ...";
```

just as if we were setting a variable.

Start of Chapter 14: Adaptive Text and Responses
Back to §14.10. Responses
Onward to §14.12. The RESPONSES testing command
Example 252: **Responsive**

Altering the standard inventory text for when the player is carrying nothing.

§14.12. The RESPONSES testing command

In practice we can't change these responses unless we know what they're called. One way to find out is just to read through the extensions we're using, but that's a laborious process. A more practical answer is to type:

```
> RESPONSES
```

which replies by listing the sets of responses currently available; for example, it says that RESPONSES 1 is the set of responses for the Standard Rules. We can then type exactly that:

```
> RESPONSES 1
Standard Rules:
    block vaguely going rule response (A): "You'll have to say which
compass direction to go in."
    print the final prompt rule response (A): "> [run paragraph on]"
    ...
```

and so on. This lists all of the responses, rule by rule, along with their current texts.

Start of Chapter 14: Adaptive Text and Responses
Back to §14.11. Changing the text of responses
Onward to Chapter 15: Numbers and Equations: §15.1. How do we measure things?

Examples from Chapter 14: Adaptive Text and Responses

245 Example Fun with Participles *RB*

Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on Clark's idle activity.

Mostly the Standard Rules use verbs adapted to finite forms ("he jumped", "we take the hammer", and so on). But Inform can also produce participles to describe actions that are ongoing: "he is carrying the fedora" or "taking the hammer..."

In this example, we give non-player characters actions to perform and then have Inform dynamically describe what they're doing when the player chooses to look.

We start by establishing the idea that a verb can describe a particular action:

```
"Fun with Participles"

Section 1 - Descriptive Functionality

Describing relates various verbs to various action
names. The verb to describe means the describing
relation.

To look around is a verb. The verb look around
describes the looking action.

To stand about is a verb. The verb stand about
describes the waiting action. To look bored is a verb.
The verb look bored describes the waiting action. To
```

```
waste time is a verb. The verb waste time describes the
waiting action.


To jump is a verb. To leap is a verb. To pirouette is a
verb. The verb jump describes the jumping action. The
verb leap describes the jumping action. The verb
pirouette describes the jumping action.
```

Now we need to give every character some sort of idle activity. By default, we'll have people just be waiting, but allow for that idle activity to change into something more interesting if the player has told them to do something else.

```
A person has an action name called the current idle.
The current idle of a person is usually the waiting action.


Rule for writing a paragraph about someone (called
chosen person) when a verb describes the current idle
of the chosen person:
    say "[The chosen person] [are] here, [present
participle of a random verb that describes (the current
idle of the chosen person)]."


Instead of someone doing something:
    now the current idle of the person asked is (the action
name part of the current action);
    continue the action.


A persuasion rule:
    persuasion succeeds.


Section 2 - Scenario


Lab is a room. The fedora is a wearable thing in the
```

```
Lab. Clark is a man in the Lab.
```

And just to give past participles a test-drive as well, let's make Clark a bit of a drama king:

```
After Clark doing something when a verb describes (the
action name part of the current action):
    say "'Fine, have it your way!' Clark exclaims. 'But I
have [past participle of a random verb that describes
(the action name part of the current action)] for the last
time!'";
    rule succeeds.


Test me with "look / Clark, jump / look / Clark, look / look
/ Clark, wait".
```

ExampleVariety *RB*

Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

Verbs can be related to other things by relations. We've seen that it's possible for a verb to "mean" a relationship. But we can also create a relation between verbs and actions. For instance, we can tell Inform that "take", "get", and "acquire" are all valid ways to describe the action of taking, and then allow it to pick a verb randomly to describe whatever action just occurred.

```
"Variety"


Section 1 - Descriptive Functionality


Describing relates various verbs to various action
names. The verb to describe means the describing
```

relation.

To take is a verb. To acquire is a verb. To get is a verb.

The verb take describes the taking action. The verb acquire describes the taking action. The verb get describes the taking action.

To drop is a verb. To put down is a verb. To discard is a verb. The verb drop describes the dropping action. The verb put down describes the dropping action. The verb discard describes the dropping action.

To sniff is a verb. To smell is a verb. The verb sniff describes the smelling action. The verb smell describes the smelling action.

To jump is a verb. To leap is a verb. To pirouette is a verb. The verb jump describes the jumping action. The verb leap describes the jumping action. The verb pirouette describes the jumping action.

After an actor doing something when the noun is nothing and a verb describes (the action name part of the current action) (this is the apply random verbs to describing nounless actions rule):
    say "[The actor] [verb rendering applied to a random verb that describes (the action name part of the current action)].";
    rule succeeds.

After an actor doing something to something when a verb describes (the action name part of the current action) (this is the apply random verbs to describing

```
actions rule):
    say "[The actor] [verb rendering applied to a random
verb that describes (the action name part of the current
action)] [the noun].";
    rule succeeds.



To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".



Section 2 - Scenario



Lab is a room. The table is here. The bat and the ball
are on the table.



Test me with "get ball / drop ball / get bat / drop bat /
smell ball".
```

---

247 **ExampleVariety 2** *RB*

This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

Some of our default actions establish relations between items in the world, and reporting on the relation ("You are now carrying the fedora") can be a valid response alongside reporting on the action itself ("You take the fedora").

To do this, we need to teach Inform explicitly which relations are the results of actions, then check this when reporting on actions:

```
"Variety 2"
```

Section 1 - Descriptive Functionality

Describing relates various verbs to various action names. The verb to describe means the describing relation.

Table of Action Results

relation	action
the taking relation	the taking action
the wearing relation	the wearing action
the taking off relation	the taking off action

To take is a verb. To acquire is a verb. To get is a verb.

The verb take describes the taking action. The verb acquire describes the taking action. The verb get describes the taking action.

To drop is a verb. To put down is a verb. To discard is a verb. The verb drop describes the dropping action. The verb put down describes the dropping action. The verb discard describes the dropping action.

To sniff is a verb. To smell is a verb. The verb sniff describes the smelling action. The verb smell describes the smelling action.

To jump is a verb. To leap is a verb. To pirouette is a verb. The verb jump describes the jumping action. The verb leap describes the jumping action. The verb pirouette describes the jumping action.

```
To don is a verb. The verb don describes the wearing
action.


To doff is a verb. The verb doff describes the taking off
action.


After an actor doing something when the noun is
nothing and a verb describes (the action name part of
the current action) (this is the apply random verbs to
describing nounless actions rule):
    say "[The actor] [verb rendering applied to a random
verb that describes (the action name part of the current
action)].";
    rule succeeds.


After an actor doing something to something when a
verb describes (the action name part of the current
action) (this is the apply random verbs to describing
actions rule):
    let current action name be the action name part of the
current action;
    if a random chance of 1 in 2 succeeds and the current
action name is a related action listed in the Table of
Action Results:
        choose a row with the related action of current
action name in the Table of Action Results;
        let R be the relation entry;
        let subject be the actor;
        let chosen object be the noun;
        say "[The subject] [are] now [present participle of a
random verb that means R] [the chosen object].";
    else:
        say "[The actor] [verb rendering applied to a
random verb that describes (the action name part of the
current action)] [the noun].";
    rule succeeds.


To decide which text is the rendering of (V - verb) (this
is verb rendering):
```

```
    decide on "[adapt V]".

To say infinitive of (V - a verb): (- {V}(1); -).

To say past participle of (V - a verb): (- {V}(2); -).

To say present participle of (V - a verb): (- {V}(3); -).

Section 2 - Scenario

Lab is a room. The fedora is a wearable thing in the
Lab.

Test me with "wear the fedora / take off the fedora /
wear fedora / take off fedora".
```

ExampleNarrative Register *RB*

Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

As we saw in "Variety", we can associate verbs with particular actions and call them up as needed. If we do that, though, we can also store additional information about those verbs and use that information to select the ideal verb to use in a particular situation.

In this example, we create a table of verbs and their meanings, together with some connotative information. Each time we report an action, we then score all the available verbs to decide which is the most suitable to use at the moment. This allows us to change the narrator's diction change mid-game and have the action descriptions change as well.

Moreover, because we're using adaptive verbs, these responses will automatically inflect properly even if we change the story tense and viewpoint.

"Narrative Register"

Section 1 - Descriptive Functionality

Describing relates various verbs to various action names. The verb to describe means the describing relation.

To take is a verb. To acquire is a verb. To get is a verb. To gain is a verb. To obtain is a verb. To pick up is a verb. To bag is a verb. To procure is a verb. To score is a verb. To grab is a verb. To snag is a verb. To snatch is a verb. To collect is a verb.

To drop is a verb. To put down is a verb. To discard is a verb. To throw away is a verb. To dispose of is a verb. To set down is a verb. To toss aside is a verb. To ditch is a verb. To abandon is a verb. To dump is a verb. To jettison is a verb. To abjure is a verb. To foresake is a verb. To dispense with is a verb.

After an actor doing something to something when a verb describes (the action name part of the current action) (this is the apply random verbs to describing actions rule):
    score the relevant verbs;
    sort the Table of Verb Meanings in reverse relevance order;
    choose row 1 in the Table of Verb Meanings;
    let top score be the relevance entry;
    sort Table of Verb Meanings in random order;
    repeat through the Table of Verb Meanings:

```
        if relevance entry is top score:
            say "[The actor] [verb rendering applied to (word
entry)] [the noun].";
            erase relevance;
            rule succeeds.



To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".



To score the relevant verbs:
    repeat through the Table of Verb Meanings:
        if the meaning entry is (the action name part of the
current action):
            increase relevance entry by 1;
            repeat with chosen connotation running through
connotations entry:
                if the chosen connotation is listed in the
current register:
                    increase relevance entry by 1;
                otherwise:
                    decrease relevance entry by 1.



To erase relevance:
    repeat through Table of Verb Meanings:
        now relevance entry is 0.



A tonality is a kind of value. The tonalities are pompous,
archaic, slangy, upbeat, downbeat.



Connoting relates various verbs to various tonalities.
The verb to connote means the connoting relation.



The current register is a list of tonalities that varies. The
current register is { }.
```

```
When play begins:
    repeat through the Table of Verb Meanings:
        now the word entry describes the meaning entry;
        now relevance entry is 0;
        repeat with chosen tone running through the connotations entry:
            now the word entry connotes the chosen tone.
```

Table of Verb Meanings

word	meaning	connotations	relevance (a number)
the verb take	taking action	{}	
the verb capture	taking action	{pompous}	
the verb get	taking action	{}	
the verb gain	taking action	{}	
the verb obtain	taking action	{pompous}	
the verb pick up	taking action	{}	
the verb bag	taking action	{slangy}	
the verb snap up	taking action	{slangy, vulgar}	
the verb procure	taking action	{archaic}	
the verb grab	taking action	{slangy}	
the verb snag	taking action	{slangy}	
the verb snatch	taking action	{slangy}	
the verb collect	taking action	{}	
the verb discard	dropping action	{pompous}	
the verb drop	dropping action	{}	
the verb put down	dropping action	{}	
the verb toss aside	dropping action	{}	
the verb ditch	dropping action	{slangy}	
the verb throw away	dropping action	{}	
the verb dispose of	dropping action	{}	
the verb set down	dropping action	{}	
the verb abandon	dropping action	{downbeat}	
the verb dump	dropping action	{downbeat}	
the verb abjure	dropping action	{archaic}	
the verb forsake	dropping action	{archaic}	
the verb jettison	dropping action	{pompous}	
the verb dispense with	dropping action	{pompous}	

Section 2 - Changing Tone Mid-Game

```
Understand "new tone" as changing the tone. Changing
the tone is an action out of world.



Carry out changing the tone:
    now the current register is { };
    if a random chance of 1 in 4 succeeds:
        say "Your narrator will now adopt an ordinary
tone.";
        rule succeeds;
    let rando be a random tonality;
    add rando to the current register, if absent;
    say "Your narrator will now be [rando]."



Section 3 - Scenario



Lab is a room. The table is here. The bat and the ball
are on the table.



Test me with "get ball / drop ball / get bat / drop bat /
new tone / get all / drop all / new tone / get all / drop all".
```

ExampleOlfactory Settings *RB*

Some adaptive text for smelling the flowers, or indeed, anything else.

While this isn't very interesting as IF, it runs through most of the adaptive-text tricks.

```
"Olfactory Settings"



The Doghouse is a room. "Not so much a place as a
state of being."
```

The player carries a ticket to the opera, some papers, and a bouquet of flowers. The bouquet is ambiguously plural.

Instead of eating something inedible, say "[The noun] [don't] seem likely to agree with [us] at all. [We][']d be wiser to leave [regarding the noun][them] alone."

Instead of touching something: say "[regarding the noun][Those] [are] all prickly."

Instead of smelling something: say "[Our] nose [regarding nothing][are] too weak to get much smell from [regarding the noun][those]."

Instead of smelling the bouquet: say "[regarding the noun][They]['re] lovely."

Instead of tasting something:
    say "Whew, [regarding the noun][are] [those] ever nasty!"

Test me with "x ticket / eat it / eat them / touch it / touch them / smell it / smell them / taste it / taste them / x papers / eat it / eat them / touch them / smell them / taste them / x bouquet / eat it / eat them / touch them / smell them / taste them".

ExampleHistory Lab
We create phrases such as "the box we took" and "the newspaper Clark looked at" based

on what has already happened in the story.

The examples Variety and Narrative Register show how verbs can be associated with particular actions. Here, we use the same principle so that we can report to the player what was last done to a particular object, either by the player or by someone else.

To do this, we need to use the idea of stored actions from the Advanced Actions chapter.

```
"History Lab"

Section 1 - Procedure

An object has an action called the last action.

Describing relates various verbs to various action
names. The verb to describe means the describing
relation.

To take is a verb. The verb take describes the taking
action.
To drop is a verb. The verb drop describes the dropping
action.
To look at is a verb. The verb look at describes the
examining action.
To examine is a verb. The verb examine describes the
examining action.

After an actor doing something to something:
    if a verb describes the action name part of the current
action:
        now the indefinite article of the noun is "the";
        now the last action of the noun is the current
```

```
action;
    continue the action.



After printing the name of something (called item):
    if the last action of the item is not waiting and the last
action of the item is not the current action:
        let chosen action-name be the action name part of
the last action of the item;
        let chosen actor be the actor part of the the last
action of the item;
        if a verb describes the chosen action-name:
            let the chosen verb be a random verb that
describes the chosen action-name;
            say " [if the chosen actor is the player][we][else]
[chosen actor][end if] [adapt chosen verb in past tense]";



Section 2 - Scenario



Lab is a room. It contains a box. The box contains a
newspaper. Clark is a man in the Lab.



A persuasion rule:
    persuasion succeeds.



Test me with "x box / look / x newspaper / look / clark, x
newspaper / clark, get box / clark, drop box / look / take
box / i / smell box / i".
```

Notice that smelling the box does not change the box's description because we haven't gotten around to defining a smell or sniff verb.

ExampleRelevant Relations *RB*

An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

Suppose that we wanted authors to be able to indicate which relations should or should not be included in room descriptions, and have the system dynamically honor that instruction.

Inform already knows about verbs for describing supporting, containment, carrying, and wearing, so we could write a set of instructions to handle such cases. To do this, we're using the "writing a paragraph about" activity, which is described in the chapter on activities.

```
"Relevant Relations"


Section 1 - Procedure


Rule for writing a paragraph about something (called
item):
    now the current paragraph is { };
    say "[one of][regarding item]There [are] [an item]
here[or][We] [can see] [an item] here[at random]. [run
paragraph on]";
    follow the descriptive rules for the item;
    repeat with new item running through the current
paragraph:
        now the prior named object is nothing;
        if new item is not the item:
            follow the descriptive rules for the new item;
    say paragraph break.


Rule for writing a paragraph about someone (called
chosen person):
    now the current paragraph is { };
    say "[one of][regarding chosen person][The chosen
person] [are] here[or][We] [can see] [a chosen person]
here[at random]. [run paragraph on]";
    follow the descriptive rules for the chosen person;
    repeat with new item running through the current
paragraph:
        now the prior named object is nothing;
        if new item is not the chosen person:
            follow the descriptive rules for the new item;
    say paragraph break.
```

The descriptive rules are an object-based rulebook.

Definition: a container is see-through:
    if it is transparent:
        yes;
    if it is open:
        yes;
    no.

A descriptive rule for a see-through container (called item) (this is the describe contents rule):
    describe the containment relation for item.

A descriptive rule for a supporter (called item):
    describe the support relation for item.

A descriptive rule for a person (called item):
    describe the wearing relation for the item.

A descriptive rule for a person (called item):
    describe the carrying relation for the item.

The current paragraph is a list of things that varies.

Before printing the name of something (called mentioned target) while writing a paragraph about something:
    add the mentioned target to the current paragraph, if absent.

```
To describe (R - a relation of objects) for (item - a thing):
    if a thing to which item relates by R is a thing:
        say "[The item with pronoun] [verb rendering
applied to a random verb that means R] [the list of
things to which item relates by R with indefinite articles].
[run paragraph on]"


To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".


To say (T - a thing) with pronoun:
    if T is the prior named object:
        say "[regarding T][They]";
    else:
        say "[The T]"


Section 2 - Scenario


The Space Elevator is a room. "Mercifully, there aren't
any windows. The ability to see how far up you are
would almost certainly make you ill."


The luggage rack is a supporter in the Space Elevator.
The suitcase is a closed openable container on the
luggage rack. The bouquet is on the luggage rack.


Clark is a man in the Space Elevator. Clark is carrying a
box of cupcakes. Clark is wearing a t-shirt. The
description of the box of cupcakes is "They're the latest
confection from Red Velvet Planet, the Martian bakery."


Persuasion rule: persuasion succeeds.
```

We can if we like then add alternate names for these relations that will be randomly swapped in some of the time. For instance:

```
To sport is a verb. The verb to sport means the wearing
relation.

To hold up is a verb. The verb to hold up means the
support relation.

Test me with "clark, drop the box / look / clark, take the
suitcase / look / clark, get bouquet".
```

One might, hypothetically, imagine going even further than this and simply designating relations as either "important" or "unimportant" -- perhaps changing the relation's designation at runtime. Relations are not themselves allowed to have properties, however.

---

252 Example **Responsive** *RB*

Altering the standard inventory text for when the player is carrying nothing.

The most straightforward way to alter the response text for something in the standard rules is to select the Index tab, then select Actions, then pick the particular action whose text we'd like to alter. Under action details, there will be icons that look like speech bubbles.

Clicking on the speech bubble will show what the current response text is, and give us an option called "set".

If we click "set", this will automatically paste in the response name that we need to change. We can put this inside a "when play begins" rule in order to make that change take effect from the start of the game, like so:

```
"Responsive"
```

An Anonymous B613 Cell is a room. "There isn't much to see in this bare room. What there is, you've already seen sometime in the last twenty years."

When play begins:
    now print empty inventory rule response (A) is "[We] [have] absolutely nothing.".

Test me with "i".

Chapter 15: Numbers and Equations

Scaling and equivalents; *§15.11. Named notations*; *§15.12. Making the verb "to weigh"*; *§15.13. The Me*

Contents of *Writing with Inform*

Chapter 14: Adaptive Text and Responses

Chapter 16: Tables

Indexes of the examples

§15.1. How do we measure things?

In a poem, or in a novel, exact scientific measurements are not the point. So a writer who wants to set up ways to describe the sky at different times might go for something like this:

```
The sky can be cadmium, mackerel, overcast or cornflower.
```

And nobody is interested in the sun angle, the percentage of cloud cover, or any of the other numbers behind all of this. Similarly, if we walk into a familiar office which has been disturbed, we might well say "Look! The filing cabinet is in the middle of the floor." We are not likely to exclaim "Look! The filing cabinet is 1.2m from the east wall and 2.1m from the north wall."

But some writers of interactive fiction do like to make use of physical realism. For instance, it's easier to forbid a bulky object being taken through a narrow doorway if there is a way to measure and compare sizes.

Most computer programs write numbers in the same way, whatever they're used for. But human beings don't. If someone says "How far is Duluth?", we're more likely to say "100 miles" than just "100". This is a useful feature of natural language, because it means we always know how to translate that number into reality - it's 100 miles, not 100 km, or 100 inches; and it's definitely a distance, not 100 apples or 100 kilograms.

Inform lets us use plain numbers if we want to, but it also allows us to create numerical kinds of value:

```
A distance is a kind of value. 5 miles specifies a distance.
```

That kind of definition, and the consequences, will be the subject of this chapter. But we will first look a little harder at the two numerical kinds of value we get for free: "number" and "real number".

Start of Chapter 15: Numbers and Equations

Back to Chapter 14: Adaptive Text and Responses: §14.12. The RESPONSES testing command

Onward to §15.2. Numbers and real numbers

§15.2. Numbers and real numbers

Inform uses two different kinds of numerical quantity: "number" and "real number". Neither is better than the other: they're different approaches, each good for a different purpose.

What Inform calls a "number" is a whole number, positive, negative or zero. The range of numbers we can hold is not unlimited - if the format Setting for a project is the Z-machine, then we have:

```
-32768, -32767, ..., -3, -2, -1, 0, 1, 2, 3, ..., 32767
```

and if it is set to Glulx, then we have:

```
-2147483648, -2147483647, ..., -3, -2, -1, 0, 1, 2, 3, ..., 2147483647
```

Numbers from zero to twelve may be written out, but larger ones must be written as numerals. So "twelve" or "12", but "13" only.

If we're using Glulx, Inform also has "real numbers" such as

```
2.1718, 4.0, -1633.9
```

which are not restricted to whole numbers, but which are stored only approximately: only about six to nine decimal digits can be relied on. For example,

```
showme 1.2345654321;
showme 1.2345667890;
```

produces

```
real number: 1.23457
real number: 1.23457
```

because these two numbers are so close together that Inform can't tell them apart. But we do also get the ability to represent enormously large or small quantities, and to help with that, Inform can read and write "scientific notation". For example,

```
let Avogadro's number be 6.022141 x 10^23;
```

is equivalent to typing

```
let Avogadro's number be 602214100000000000000000.0;
```

The "x 10^23" part tells Inform that the decimal point belongs 23 places to the left of where it's written. (In scientific papers, the 23 would be printed as a superscript -- it's 10 to the power 23 -- but that's not convenient to type in to the source text, so we use the "^" symbol to indicate superscript.) The range we can hold is roughly:

```
1.18 x 10^–38 to 3.4 x 10^38
```

It's hard to convey just how enormously different these two numbers are: if we used them to measure widths in meters, one would be a hundred trillion trillion times smaller than an atom, the other a billion times larger than the entire visible universe. Scientific notation is the ultimate adjustable spanner.

Inform also allows the two most famous real numbers in mathematics to be given by their names:

```
pi
e
```

which are close to 3.14159265 and 2.7182818 respectively. (Lower case letters must be used: these can't be written "Pi" or "E". Euler's constant gamma, always in the bronze medal position, will have to be written out longhand as 0.5772156649.)

Most computer programming languages traditionally write floating-point numbers using the E notation, like so:

```
6.022141E+23;
```

Inform will follow suit if the use option "Use engineering notation." is set, but by default it isn't.

Start of Chapter 15: Numbers and Equations
Back to §15.1. How do we measure things?
Onward to §15.3. Real number conversions
Example 253: ★★★ **Alias** A telephone with phone numbers of the standard American seven-digit length.

§15.3. Real number conversions

This section notes down some technicalities about real numbers which need to be put down in writing somewhere, but won't affect most people most of the time.

Inform allows us to use numbers whenever real numbers are expected, and converts them automatically. For example,

```
cosine of 2
```

is read as if it were

```
cosine of 2.0
```

and produces -0.41615 either way. This conversion goes from exactness to

approximation, so we may lose a little accuracy: real numbers measure to an accuracy of about 1 part in 16000000, so they'll have trouble telling the difference between 16000000 and 16000001. But this is unlikely to matter, since real numbers are used only for approximate calculations anyway.

The ordinary arithmetic operations work on both numbers and real numbers, so the meaning of "N plus M" depends on the kinds of N and M. In general the rule is that if either is a real number then the other one is automatically converted, and real arithmetic is used. So:

```
3 divided by 2 = 1
3 divided by 2.0 = 1.5
3.0 divided by 2 = 1.5
3.0 divided by 2.0 = 1.5
```

In general we can't do the reverse, that is, we can't silently use a real number where a number is expected. For example,

```
word number 1.6 in "The Great Wall of China"
```

makes no sense. But we can explicitly convert them:

(real number) **to the nearest whole number ... number**

This phrase performs signed addition on the given values, whose kinds must agree, and produces the result. Examples:

```
1.4 to the nearest whole number = 1
1.6 to the nearest whole number = 2
-1.6 to the nearest whole number = -2
```

We probably ought to bear in mind that the limited range of "number" means that the nearest whole number might not be all that near. For example:

```
6 x 10^23 to the nearest whole number = 2147483647
```

because 2147483647 is the highest value a "number" can have.

Finally, real number can also store two interesting not-really-number sorts of value. First, we have

```
plus infinity, minus infinity
```

which are used to keep track of what happens when we divide by really small quantities. It's mathematically impossible to divide by 0, but this can be hard to avoid when we're using real numbers, because they're only approximately stored - so it's not always possible to say whether they're exactly 0 or not. So in real number arithmetic,

```
showme 1.0 divided by 0.0;
```

doesn't throw a run-time problem the way that

```
showme 1 divided by 0;
```

does. Instead, it produces "plus infinity". Infinity behaves roughly the way we might expect - for example, "2 divided by plus infinity" produces 0 - but once it comes into a calculation the result probably lies on some extreme and won't be very useful. Amusingly, the following is correct Inform syntax:

```
plus infinity to the nearest whole number
```

and evaluates of course to 2147483647. We can use the adjectives "infinite" and "finite" to talk about these numbers: plus infinity and minus infinity are infinite, everything else is finite.

The same problem occurs for calculations like square roots. It's impossible to take the square root of a negative number, but we don't want to throw a run-time problem, because approximation means we can't always guarantee to stay the right side of 0. So for a few calculations like this, Inform generates what's called a "nonexistent" real number. We can use the adjectives nonexistent or existent to talk about this. Every number mentioned on this page so far is "existent", including the infinities. The only way to get a nonexistent number is to carry out an impossible mathematical operation such as

```
logarithm of -10
```

(The design of "real number" here follows well established trade-offs for scientific computing. Inform follows the IEEE-754 binary32 standard for floating-point arithmetic, so Inform's "real number" behaves very like the "float" type in C, C++, Java and similar programming languages. A "nonexistent" number is what's often called a NaN - a Not-a-Number.)

Start of Chapter 15: Numbers and Equations

Back to §15.2. Numbers and real numbers

Onward to §15.4. Printing real numbers

§15.4. Printing real numbers

say "[(real number) **to** (number) **decimal places]"**

This text substitution writes out the number to the given number of decimal places. Examples:

```
"The semicircle is roughly [pi to 3 decimal places] paces
around."
```

produces "The semicircle is roughly 3.142 paces around." The number of places can only usefully be from 1 to 8. Note that, for example, "[1.235 x 10^-7 to 3 decimal places]" produces 0.0; "[1.235678 x 10^8 to 3 decimal places]" produces "1.236 × 10^8".

say "[(real number) **in decimal notation]"**

This text substitution writes out the number in decimal form, that is, avoiding "x 10^n" even for very large or very small quantities. For example,

```
"[1.23457 x 10^8 in decimal notation]"
```

produces 123457000.0 rather than 1.23457 x 10^8. This can look pretty

extreme: for example, "[1.8983 x 10^27 in decimal notation]", the mass of the planet Jupiter in kilograms, produces 1898296960000000000000000000.0.

say "[(real number) **to** (number) **decimal places in decimal notation]"**

This text substitution writes out the number in decimal form, but rounding to the accuracy given.

say "[(real number) **in scientific notation]"**

This text substitution writes out the number in scientific form, that is, using "x 10^n" even for easy-to-judge quantities. For example,

"[the reciprocal of 137 in scientific notation]"

produces 7.29927 x 10^-3 rather than 0.0073. This can look odd: for example, "[pi in scientific notation]" comes out as 3.14159 × 10^0 rather than 3.14159.

say "[(real number) **to** (number) **decimal places in scientific notation]"**

This text substitution writes out the number in scientific form, but rounding to the accuracy given.

Start of Chapter 15: Numbers and Equations
Back to §15.3. Real number conversions
Onward to §15.5. Arithmetic

§15.5. Arithmetic

We are allowed to perform about the same operations on numbers as are provided by a simple office calculator, starting with addition, subtraction, multiplication and

division. We can use the traditional typewriter symbols for these, +, -, * and /, or can spell them out in words as "plus", "minus", "times" (or "multiplied by"), and "divided by". Definitively:

(arithmetic value) **+** (arithmetic value) **... value**

or:
(arithmetic value) **plus** (arithmetic value) **... value**

This phrase performs signed addition on the given values, whose kinds must agree, and produces the result. Examples:

```
200 + 1 = 201
10:04 AM + two minutes = 10:06 AM
```

(arithmetic value) **-** (arithmetic value) **... value**

or:
(arithmetic value) **minus** (arithmetic value) **... value**

This phrase performs signed subtraction on the given values, whose kinds must agree, and produces the result. Examples:

```
200 - 1 = 199
10:04 AM - two minutes = 10:02 AM
```

(arithmetic value) ***** (arithmetic value) **... value**

or:
(arithmetic value) **times** (arithmetic value) **... value**

or:
(arithmetic value) **multiplied by** (arithmetic value) **... value**

This phrase performs signed multiplication on the given values, whose kinds

must be dimensionally compatible, and produces the result. Examples:

```
201 times 3 = 603
two minutes times 4 = eight minutes
```

(arithmetic value) / (arithmetic value) **... value**

or:
(arithmetic value) **divided by** (arithmetic value) **... value**

This phrase performs signed division on the given values, whose kinds must be dimensionally compatible, and produces the result. Examples:

```
201 divided by 3 = 67
202 divided by 3 = 67
202.0 divided by 3 = 67.33334
twenty minutes divided by 4 = five minutes
twenty minutes divided by five minutes = 4
```

Division rounds whole-number values down to the nearest whole number. An attempt to divide a number by 0 will cause a run-time problem message; but an attempt to divide a real number by 0 will instead produce plus infinity or minus infinity.

remainder after dividing (arithmetic value) **by** (arithmetic value) **... value**

This phrase performs signed division on the given values, whose kinds must be dimensionally compatible, and then produces the remainder. Examples:

```
remainder after dividing 201 by 5 = 1
remainder after dividing twenty minutes by 7 = six minutes
```

It is mathematically impossible to divide by 0, so any attempt to find the remainder after dividing a number by 0 will cause a run-time problem message. For a real number this won't arise and the remainder will usually be 0.0.

The verbal and symbolic forms of these phrases are equivalent:

```
the score + 10
the score plus 10
```

It's probably better style to spell them out in full when writing text, and keep the symbols for writing equations, as we'll see later on in the chapter. (If we do use the symbols, then spaces around them are obligatory: to Inform, they are words which just happen to be spelt with symbols instead of letters.)

Arithmetic often produces fussily exact answers which seem inappropriate in a conversation. Nobody says "Steeple Barton is 7.655 miles down the road", but "Steeple Barton is eight miles down the road" sounds perfectly normal. In order to make that sort of report easier to make, Inform provides another arithmetic operation, one that's not found in most computer programming languages:

(arithmetic value) **to the nearest** (arithmetic value) **... value**

This phrase rounds the given value off, rounding upward in boundary cases. Examples:

```
201 to the nearest 5 = 200
205 to the nearest 10 = 210
10:27 AM to the nearest five minutes = 10:25 AM
```

Inform has very few mathematical functions built in as phrases, because these aren't very often needed in story-telling. But it does provide these:

square root of (arithmetic value) **... value**

This phrase produces an approximate square root, to the nearest integer, of the given value, which must be of a kind which has square roots. Example:

```
square root of 16 = 4
```

Trying to take the square root of a negative number will cause a run-time problem, because then we can't even nearly solve it.

(Warning: this is slow to compute if the Z-machine setting is used. For

best performance, use Glulx.)

real square root of (arithmetic value) **... value**

This phrase produces a square root, as accurately as a real number can hold it, of the given value, which must be of a kind which has square roots. Example:

```
real square root of 2 = 1.41421
```

The real square root of a negative number is nonexistent.

cube root of (arithmetic value) **... value**

This phrase produces an approximate cube root, to the nearest integer, of the given value, which must be of a kind which has cube roots. Example:

```
cube root of 27 = 3
cube root of -27 = -3
```

(Warning: this is not very accurate if the Z-machine setting is used. For

best performance, use Glulx.)

We can compare numbers using either the traditional computer-programming symbols, or using words:

```
if the score is less than 10
if the score < 10
```

and similarly for "greater than", "at least" and "at most", with the symbols ">", ">=" and "<=". But we are not allowed the equals sign: for that we need only use "is" -

```
if the score is 10
```

§15.6. Powers and logarithms

If the last section provided a basic office calculator, this section and the next provide the more exotic rows of buttons found on a scientific calculator. All of these are done using real number arithmetic. To start with some dull ones, here are two ways to round off numbers:

ceiling of (real number) **... real number**

Produces the smallest integer value greater than or equal to the one given. Examples:

```
ceiling of pi = 4.0
ceiling of -16.315 = -16.0
```

(Note that the result is still a real number; it simply has no fractional part any more.)

floor of (real number) **... real number**

Produces the largest integer value less than or equal to the one given. Examples:

```
floor of pi = 3.0
floor of -16.315 = -17.0
```

(Note that the result is still a real number; it simply has no fractional part any more.)

Two more easy functions:

absolute value of (real number) **... real number**

Removes the sign from a value, leaving positive numbers alone but making negative ones positive. Examples:

```
absolute value of 62.1 = 62.1
absolute value of 0 = 0.0
absolute value of -62.1 = 62.1
absolute value of minus infinity = plus infinity
```

reciprocal of (real number) **... real number**

Calculates 1/x, that is, divides up 1 into this many pieces. Examples:

```
reciprocal of -2 = -0.5
reciprocal of 0.1 = 10.0
reciprocal of 7 = 0.14286
reciprocal of plus infinity = 0.0
```

Now for taking powers. In general we have:

(real number) **to the power** (real number) **... real number**

Computes x to the power y. Examples:

```
2 to the power 4 = 16.0
100 to the power 0.5 = 10.0
7 to the power -1 = 0.14286
pi to the power 0 = 1.0
```

In the words of the Glulx specification document (section 2.12), "the special cases are breathtaking": if you need to know exactly what, say, "minus infinity to the power Y" will do for different cases of Y, refer to the details of the "pow" opcode.

To compute square roots, it's more efficient to use "real square root of X" function than "X to the power 0.5", though both work. To obtain the Nth root of X, we might use:

```
X to the power (reciprocal of N)
```

being careful to use "reciprocal of N" rather than "1 divided by N" to make sure we're using real and not integer arithmetic.

Similarly, the following is more efficient than "e to the power ...", but equivalent to it:

exponential of (real number) **... real number**

Computes e to the given power, where e is the base of natural logarithms. Examples:

```
exponential of 0 = 1.0
exponential of 1 = e = 2.7182818
exponential of -10 = 4.53999 x 10^-5
exponential of 10 = 22026.46484
exponential of logarithm of 7.12 = 7.12
```

The reverse of taking powers is taking logarithms.

logarithm to base (number) **of** (real number) **... real number**

Finds what power the base would have to be raised to in order to get this value. Examples:

```
logarithm to base 10 of 1000000 = 6.0
logarithm to base 10 of 350 = 2.54407
logarithm to base 2 of 256 = 8.0
```

Logarithms of zero or negative numbers are nonexistent. Note that "logarithm to base 10 of ..." is what most calculators call simply "log", but Inform doesn't: it uses "log" for natural logarithms.

natural/-- **logarithm of** (real number) **... real number**

Finds what power e would have to be raised to in order to get this value. Examples:

```
logarithm of e = 1.0
logarithm of 1 = 0.0
logarithm of 1000 = 6.90776
logarithm of exponential of 7.12 = 7.12
```

Logarithms of zero or negative numbers are nonexistent. This is the function which most calculators label as "ln", for "log natural", but in mathematical and scientific papers it's more often written "log", and Inform follows that convention.

Start of Chapter 15: Numbers and Equations
Back to §15.5. Arithmetic
Onward to §15.7. Trigonometry

§15.7. Trigonometry

We have twelve functions left to cover, though they are all closely related.

(real number) **degrees ... real number**

Inform measures angles in radians, a convention in which the angle for a half circle is pi, and a right angle is pi divided by 2. This is better from a mathematical point of view, but in practice most people think about angles using degrees, where 180 degrees is a half-circle and a right angle is 90 degrees. This phrase helps with that by converting from degrees to radians: in other words, it multiplies by 0.0174532925, since that's roughly 1/180th of pi. Examples:

```
sine of 90 degrees = 0.0
cosine of 60 degrees = 0.5
```

sine of (real number) **... real number**

The length of the upright of a right-angled triangle with this angle and a hypotenuse of length 1, where angle is measured in radians. Examples:

```
sine of 0 = 0
sine of 45 degrees = 0.70711
sine of (pi divided by 4) = 0.70711
sine of (pi divided by 2) = 1.0
sine of pi = 0
```

cosine of (real number) **... real number**

The length of the base of a right-angled triangle with this angle and a hypotenuse of length 1, where angle is measured in radians. Examples:

```
cosine of 0 = 1.0
cosine of 45 degrees = 0.70711
cosine of (pi divided by 4) = 0.70711
cosine of (pi divided by 2) = 0.0
cosine of pi = -1.0
```

tangent of (real number) **... real number**

The ratio of the upright length to the base length in a right-angled triangle with this angle and a hypotenuse of length 1, where angle is measured in radians. Examples:

```
tangent of 0 = 0.0
tangent of 45 degrees = 1.0
tangent of (pi divided by 4) = 1.0
tangent of (pi divided by 2) = plus infinity
```

arcsine of (real number) **... real number**

The inverse of the sine function.

arccosine of (real number) **... real number**

The inverse of the cosine function.

arctangent of (real number) **... real number**

The inverse of the tangent function.

hyperbolic sine of (real number) **... real number**

The hyperbolic sine function, often written "sinh" but pronounced "shine".

hyperbolic cosine of (real number) **... real number**

The hyperbolic cosine function, often written "cosh".

hyperbolic tangent of (real number) **... real number**

The hyperbolic tangent function, often written "tanh".

hyperbolic arcsine of (real number) **... real number**

The inverse of the hyperbolic sine function.

hyperbolic arccosine of (real number) **... real number**

The inverse of the hyperbolic cosine function.

hyperbolic arctangent of (real number) **... real number**

The inverse of the hyperbolic tangent function.

Start of Chapter 15: Numbers and Equations

Back to §15.6. Powers and logarithms

Onward to §15.8. Units

§15.8. Units

Suppose we want to talk about how tall people are. We could just create a "number" property, like this:

```
A person has a number called height.
```

But then we would have to write lines like "Isabella has height 68", which nobody would naturally say. What we want is to be able to write "Isabella is 5 foot 8." Perhaps the computer will need to store that measurement as the number 68 in some register or other, but we don't want to know about that.

"5 foot 8" is a complicated notation in a way - it involves both feet and inches - so let's start with a simpler example:

```
A weight is a kind of value. 10kg specifies a weight.
```

This is a little different to the kinds of value seen so far, which were all created like so:

```
A colour is a kind of value. The colours are red, green and blue.
```

We can't mix the two styles: a new kind of value will either be numerical at heart ("10kg") or verbal at heart ("blue").

The effect of "10kg specifies a weight" is to tell Inform that this is the notation for writing a constant "weight". So, for instance,

```
The maximum load is a weight that varies. The maximum load is
8000kg.
```

```
if the maximum load is greater than 8000kg, ...
```

Inform is then careful not to allow weights to be mixed up with other numerical values. For instance, it won't allow "if the maximum load is 400", because 400 is a number, not a weight.

More or less anything we can do with numbers, we can now do with weights. For instance, we can write:

```
The Weighbridge is a room. "A sign declares that the maximum load
is [maximum load]."
```

...which will produce the text "A sign declares that the maximum load is 8000kg."

Numerical kinds of value are sometimes called "units", because one of their main uses is to allow us to write quantities using scientific units such as kilograms. But they have other uses too. We have a great deal of freedom in creating notations like "10kg", or "4 foot 10" - the main thing is that new notations must not already mean a value. So "10 specifies a weight" will not be allowed, because 10 specifies a number already.

By default we can only write whole-number values. As we've seen, Inform can handle both integer (whole-number) and real arithmetic, and they each have their advantages. The default here is to use whole numbers, so

```
10 kg specifies a weight.
```

will store only whole numbers of kilograms (unless clever scaling tricks are used: see the next section). That may be fine, but if we need to handle a wider range of weights, or do scientific calculations that need to be more accurate, this is better:

```
1.0 kg specifies a weight.
```

Here Inform can see from the decimal point in the prototype number that real numbers will be involved. We can still write "8000kg", but we can now also write

"1.9885 x 10^30 kg" (the mass of the Sun) or "9.109383 x 10^–31 kg" (the mass of an electron). On the other hand, any calculations we do will be limited in accuracy to about 6 to 9 decimal places, exactly as for real numbers.

By default we can only write positive values when whole numbers are used. Sometimes it is unnatural to write negative values, and so Inform will issue a Problem message if this is tried - for instance, Inform would not allow us to write a weight of -4 kg. (This doesn't mean that arithmetic on units is forbidden to get a negative result: we may want to work out the difference between two weights. Inform's Problem message is simply to try to prevent the accidental writing of incorrect values.) If we do want the ability to write negative values in the source text, we signal that in the notation itself:

```
-10 kg specifies a weight.
```

That alerts Inform that both positive and negative values for this unit make sense.

If we set up a spread of multiple notations (see the next section) then this is automatically enabled, because then we're clearly dealing with proper physics, where negative values are common; and similarly if we use real numbers (as above).

Start of Chapter 15: Numbers and Equations

Back to §15.7. Trigonometry

Onward to §15.9. Multiple notations

Example 254: **rBGH** The player character's height is selected randomly at the start of play.

Example 255: **Lethal Concentration 1** A poisonous gas that spreads from room to room, incapacitating or killing the player when it reaches sufficient levels.

Example 256: **Wonderland** Hiking Mount Rainier, with attention to which locations are higher and which

lower than the present location.

Example 257: **Lethal Concentration 2** Poisonous gas again, only this time it sinks.

---

§15.9. Multiple notations

Going back to our weight example:

```
A weight is a kind of value. 10kg specifies a weight.
```

The notation here is a single word, even if it contains digits as well as letters - "10kg". But it doesn't have to be one word. These would have worked, too:

```
10kg net specifies a weight.
10 kg specifies a weight.
```

In fact, we are allowed to have all three at once, as alternatives:

```
A weight is a kind of value. 10kg specifies a weight. 10kg net
specifies a weight. 10 kg specifies a weight.
```

If we often have to deal with large weights, it becomes a little cumbersome to keep on writing something like "80000kg". An engineer would write "80 tonnes" for this. Similarly, we wouldn't like road maps to use light years, or speed limit signs to use furlongs per fortnight. So it's sometimes useful to provide a spread of different notations, at different scale factors, for the same kind of value. Here's one way of setting up the tonne, that is, the metric ton:

```
1 tonne specifies a weight scaled up by 1000.
```

This really is an alternative way to write the same thing: for instance, Inform will allow "25kg plus 3 tonne", the result being "3.025 tonne".

That's all very well, but a value like "3 tonne" reads a little oddly, even if it's correct in theory. Outside of scientific journals with old-school copy editing, most people would write "3 tonnes", not "3 tonne". Here's a better try:

```
1 tonne (singular) specifies a weight scaled up by 1000.
2 tonnes (plural) specifies a weight scaled up by 1000.
```

Now Inform will not only recognise both forms, but also use the right one when printing back.

Start of Chapter 15: Numbers and Equations
Back to §15.8. Units
Onward to §15.10. Scaling and equivalents

§15.10. Scaling and equivalents

As we've seen, there are two ways to store values like lengths or weights: as whole numbers, or as real numbers. If we prefer to use whole numbers, or if real numbers aren't available (for example if we're using the Z-machine setting), then we might run into an awkward problem: when we write

```
1 kg specifies a weight.
```

we make this correspond to the whole number "1", and that means Inform can never handle weights smaller than 1 kg.

But as we've seen, we can provide differently scaled notations for the same unit:

```
A length is a kind of value. 1m specifies a length.
1km specifies a length scaled up by 1000.
```

And this allows us to write "0.45km" instead of "450m", if we want to, both having the same effect. "0.45km" doesn't make a real number, despite the decimal point - it's simply another way to write "450m", stored internally as the whole number 450.

Just as we can scale up, so we can also scale down:

```
1cm specifies a length scaled down by 100.
```

Now we have a spread of three notations, so "3cm", "0.03m" and "0.00003km" all mean the same thing. But something quite interesting happened at the same time: Inform realised that we want to know lengths to a greater accuracy than just a whole number of meters.

If we're using whole numbers, and we want to resolve down to very small values, that reduces the size of the largest value we can have. For instance, with the Glulx format setting, writing just

```
A length is a kind of value. 1m specifies a length.
```

gives us a range of 1m up to 2147483647m, which is plenty - it's about six times the distance from the Earth to the Moon. Going down to centimeters:

```
A length is a kind of value. 1m specifies a length. 1cm specifies a
length scaled down by 100.
```

gives us instead 1cm up to 21474836.47m, which is still enough to represent any possible distance on the Earth's surface. For instance, London to Sydney is about 17000000m.

Left to itself, Inform chooses the scaling for a unit so that it can represent exactly 1 of the smallest notation - so in our example Inform resolves down to 0.01m, not 1m, in order that it can represent 1cm accurately. But we can also fix the scaling ourselves:

```
A length is a kind of value. 1m specifies a length scaled at 10000.
```

Notice "scaled at", not "scaled down" or "scaled up" - this is now the first notation for length, so there's no existing notation which it could scale up or down. Anyway, now the range is 0.0001m, the width of a human hair, up to 214748.3647m, which is about 130 miles. (The Kinds index automatically keeps track of the range of values represented exactly.) The "scaled at" feature is meaningless if we're using real numbers, so it throws a Problem message.

Finally, for a really deluxe kind of value, we can also provide "equivalent" notations. The idea here is that we might want both miles and kilometers to work, even though they aren't direct scalings of each other. We can only do this approximately, but:

```
1 mile specifies a length equivalent to 1609m.
```

Equivalent notations are never normally used in printing values back (but see the next

section) - we wouldn't want Inform to print a sequence of values such as "1.6km", "1.65km", "1.056 miles", ... in an effort to be helpful.

Start of Chapter 15: Numbers and Equations
Back to §15.9. Multiple notations
Onward to §15.11. Named notations

§15.11. Named notations

When it has a variety of notations to choose from, Inform will normally use the neatest one given the size of the value it is printing. Suppose we've set up "weight", with three notations:

```
A weight is a kind of value. 10kg specifies a weight.
1 tonne (singular) specifies a weight scaled up by 1000.
2 tonnes (plural) specifies a weight scaled up by 1000.
```

Inform will then print back values like so:

```
45kg -> "45kg"
1000kg -> "1 tonne"
2500kg -> "2.5 tonnes"
80000kg -> "80 tonnes"
```

Note the way Inform goes into decimal places in order to talk about 2500kg in terms of tonnes rather than kilograms - it is minimising the integer part of the unit, but trying to keep it non-zero. So Inform prefers "45kg" to "0.045 tonnes".

Although Inform's habit of choosing the best notation available is usually just what we want, we sometimes want to make the choice ourselves. For instance, if we were printing out a table of different weights, we might want to give all of them in kilograms, whatever their size. In that case we can, if we want, give names to our different notations:

```
1 tonne (singular, in tonnes) specifies a weight scaled up by 1000.
2 tonnes (plural, in tonnes) specifies a weight scaled up by 1000.
```

Now we could write, for instance:

```
"The weighbridge warns you not to exceed [the maximum load in
tonnes]."
```

And the figure will always use tonnes now, even if Inform would normally think it odd: "The weighbridge warns you not to exceed 0.001 tonnes." But it will still correctly use "tonne" or "tonnes" as appropriate - what has changed is that instead of choosing from all of the weight notations, Inform now chooses from the notations labelled as "in tonnes".

Start of Chapter 15: Numbers and Equations

Back to §15.10. Scaling and equivalents

Onward to §15.12. Making the verb "to weigh"

§15.12. Making the verb "to weigh"

So now we can invent notations for weight. We could, for instance, write:

```
Weight is a kind of value. 1kg specifies a weight. Every thing has a
weight.
```

And that allows us to write:

```
The lead pig is in the Salt Mine. The weight of the lead pig is 45kg.
```

But nobody would say it that way: they'd say "The lead pig weighs 45kg." So what we really need to complete our setup is a verb "to weigh".

We have already created new verbs, but none of those methods are quite convenient for this. We want to relate something tangible (the lead pig) to something intangible (45kg), and there's no convenient relation to express this; if we set it up as a condition, we'd get something we couldn't assert, only test. Instead, we'll do something different this time:

```
The verb to weigh means the weight property.
```

Previous definitions like this ended "means the ... relation", rather than "means the ... property", but the idea is the same. The meaning of "X weighs Y" is that the weight property of X is equal to Y. So we can now write:

```
A thing usually weighs 1kg. The lead pig weighs 45kg.
something weighing 20kg
if three things weigh 5kg, ...
```

And as we saw in the chapter on Descriptions, we can also set up adjectives, comparatives and superlatives:

```
Definition: A thing is heavy if its weight is 20kg or more.
```

which creates "heavy", "heavier" and "heaviest".

Start of Chapter 15: Numbers and Equations

Back to §15.11. Named notations

Onward to §15.13. The Metric Units extension

Example 258: **Dimensions** This example draws together the previous snippets into a working implementation of the weighbridge.

Example 259: **Lead Cuts Paper** To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster.

---

§15.13. The Metric Units extension

To sum all of this up, what started out as a simple business of setting a notation for lengths becomes something quite elaborate when we try to match the actual notations used by scientists and engineers. It's all optional, of course, but as we want more and more of this, we might find ourselves with a spread of notations like this:

1mm ... 1cm ... 1m ... 1km

In addition we might want equivalents for the inch, the yard and the mile; and verbal forms like the meter and the millimeter, and then alternate spellings like the kilometre; and then both singular and plural forms. And that's just length - what about density, area, pressure, velocity and a dozen other physical quantities? After a while these declarations start to look as vastly fussy as a box of presentation cutlery.

Fortunately the whole set is indeed available in a presentation box, and at no extra charge.

(a) The built-in extension "Metric Units by Graham Nelson" sets up a whole range of scientific units, with all the notations we are likely to want. Real numbers are used throughout, so large and small-scale calculations can be carried out quite accurately. Like the other built-in extensions, it has its own documentation and examples.

(b) The built-in extension "Approximate Metric Units by Graham Nelson" does the same but using whole numbers, scaled about right for human situations. This won't be much use for extensive calculations, and won't be as accurate, but it will work reasonably well if real arithmetic isn't available.

Start of Chapter 15: Numbers and Equations
Back to §15.12. Making the verb "to weigh"
Onward to §15.14. Notations including more than one number

§15.14. Notations including more than one number

We've seen quite enough scientific notation for the time being. There are plenty of other notations used in natural language, for everyday concepts, where people don't use a tidy spread of powers of 10. Instead they use mixtures, with some sort of punctuation or text to divide them. For instance, the running time of a piece of music is easier to follow in minutes and seconds than in seconds alone: old-fashioned LP sleeves used to quote running times in the form 4'33.

```
A running time is a kind of value. 3'59 specifies a running time.
```

The choice of "3" here makes no difference, much as the choice of "10" in the weight examples was arbitrary. But the "59" is significant. Numbers after the first one are expected to range from 0 up to the value we quote - so in this case, the number of seconds can be anything from 0 to 59. Or, for instance:

```
A height is a kind of value. 5 foot 11 specifies a height.
```

A specification can contain up to eight numbers like this, but once again we might need to worry about the maximum value which can be stored. For instance, using the 3'59 notation, we can only go up to 546'07 (if we're using the Z-machine format setting) - a little over 9 hours, so the new Tori Amos album will not be a problem, but some of the more punishing German operas might break the bank.

In notations like this, only the first-appearing number part is allowed to be negative, and then only when declared with a minus sign:

```
A secret sign is a kind of value. -2x17 specifies a secret sign with
parts mystery and enigma.
```

Here, the mystery can be negative, but not the enigma.

Notations must not contain double-quotation marks because, even though people did once use these to denote minutes of arc, they would simply confuse programs like Inform's user interface which have to keep track of what is quoted text and what is not. But other punctuation marks are fine *provided they occur between two digits*. For instance, in

```
A monetary value is a kind of value. $1.99 specifies a monetary
value.
```

the full stop between the 1 and the 99 is not interpreted as a division of two sentences; and similarly for colons in examples such as

```
An aspect ratio is a kind of value. 16:9 specifies an aspect ratio.
```

Back to §15.13. The Metric Units extension
Onward to §15.15. The parts of a number specification

§15.15. The parts of a number specification

We often need to break up a number specification into its pieces. For instance, suppose we want to know the dollars part of $1.99? We can do this by naming the parts:

```
A monetary value is a kind of value. $1.99 specifies a monetary value
with parts dollars and cents.
```

We can now find the relevant parts like so. Suppose that "sum" is a monetary value. Then:

```
dollars part of sum
cents part of sum
```

are both numbers, so for instance we can

```
say "Looks like around [dollars part of sum in words] dollar[s]."
```

We can also go the other way:

```
monetary value with dollars part 4 cents part 72
```

produces the monetary value $4.72. (Note the lack of commas or "and"s, and that the parts have to be given in the right order.) This is really intended to be useful when we manipulate such values in unusual ways:

```
An aspect ratio is a kind of value. 16:20 specifies an aspect ratio with
parts width and height.
```

```
To decide which aspect ratio is the wider version of (AR - an aspect
ratio):
    let W be the width part of AR multiplied by 2;
    let H be the height part of AR;
    let the wider ratio be the aspect ratio with width part W height part
H;
    decide on the wider ratio.
```

Declaring the parts of a number specification individually also enables us to tack one or more options onto any of the parts:

```
A monetary value is a kind of value. $1.99 specifies a monetary value
with parts dollars and cents (optional, preamble optional).
```

This declares that the "cents" part is optional - it will be 0 if not specified - and that if omitted, the non-numeric "preamble" before it should also be omitted. Thus "$3" is now valid and equivalent to "$3.00": indeed it will be the preferred form when Inform prints out a monetary value which is an exact number of dollars. If we had said that "cents" was optional, but not said that the preamble was optional, then "$3." would have been the form - which is less satisfactory.

There is only one other option: "without leading zeros", as in the following.

```
An aspect ratio is a kind of value. 16:20 specifies an aspect ratio with
parts width and height (without leading zeros).
```

This ensures that when the ratio 4:3 is printed, it will be printed as "4:3" and not "4:03" as would otherwise happen.

Start of Chapter 15: Numbers and Equations
Back to §15.14. Notations including more than one number
Onward to §15.16. Understanding specified numbers

Example 260: **ZqIran Era 8** Creating an alternative system of time for our game, using new units.

---

§15.16. Understanding specified numbers

It may be worth noting in passing that number specifications, like all other kinds of value, can be understood in typed commands. (See the chapter on Understanding for more on what can go in such square brackets.) For instance:

```
"America Stands Tall"

The Oval Office is a room. Josh and Toby are men in the Oval. A
height is a kind of value. 5 foot 11 specifies a height. A person has a
height. Josh is 5 foot 8. Toby is 5 foot 10.

Height guessing is an action applying to one thing and one height.
Understand "guess [someone] is [height]" as height guessing.

Check height guessing: if the noun is not a person, say "You can only
guess the height of people." instead. Carry out height guessing: if the
height of the noun is the height understood, say "Spot on!"; if the
height of the noun is greater than the height understood, say "No,
[the noun] is taller than that."; if the height of the noun is less than the
height understood, say "No, [the noun] is shorter than that."

Test me with "guess josh is 6 foot 3 / guess josh is 5 foot 9 / guess
josh is 5 foot 3 / guess josh is 5 foot 8".
```

Start of Chapter 15: Numbers and Equations
Back to §15.15. The parts of a number specification
Onward to §15.17. Totals

Example 261: **Snip** A string which can be cut into arbitrary lengths, and then tied

§15.17. Totals

This chapter began by mentioning arithmetic, and then went on a long diversion to create scientific units, everyday weights and measures, and other notational conveniences. Putting all of that together, it's time now to calculate something with all of these numerical quantities.

Suppose we invent the idea of weight, and give everything a weight of its own. Most items will have a nominal weight of 1kg, but people will be heavier. Going on actuarial tables, we might say:

```
A weight is a kind of value. 10kg specifies a weight. Everything has a
weight. A thing usually has weight 1kg. A man usually has weight
80kg. A woman usually has weight 67kg.
```

```
Definition: A thing is light if its weight is 3kg or less.
Definition: A thing is heavy if its weight is 10kg or more.
```

and this provides us with "lighter", "lightest", "heavier" and "heaviest" as before. Now we could say "if Peter is heavier than Paul", or even "if Peter is heavier than 75kg", and so forth. We need one more tool:

total (arithmetic values valued property) **of** (description of values) **... value**

This phrase produces the total of some property held by all of the values matching the description. A problem message is produced if the values in question can't have that property ("the total carrying capacity of scenes"), or if it holds a kind of value which can't meaningfully be added up ("the total description of open doors"). Example:

```
total carrying capacity of people in the Deep Pool
```

That gives us everything we need for a working balance platform:

```
The balance platform is a supporter in the Weighbridge. "The balance
platform is currently weighing [the list of things on the platform]. The
scale alongside reads: [total weight of things on the platform]."
```

Note that this only works because we said that "everything has a weight": otherwise it would make no sense to add up the weights of things.

This enables us to get the average weight of a group of things, too:

```
the total weight of things on the platform divided by the number of
things on the platform
```

But we should be careful that this does not accidentally divide by zero, which it will if the platform has nothing on it! As well as the average, we could find the maximum and minimum weights:

```
the weight of the heaviest thing on the platform
the weight of the lightest thing on the platform
```

We should remember that "the heaviest thing on the platform" may be ambiguous, because there may be several equally heavy things there. That means

```
if the lead pig is the heaviest thing on the platform
```

will only reliably work if there is no possibility of a tie. A safer bet is:

```
if the lead pig is the weight of the heaviest thing on the platform
```

Start of Chapter 15: Numbers and Equations
Back to §15.16. Understanding specified numbers
Onward to §15.18. Equations

Example 262: ★★★ **Nickel and Dimed** A more intricate system of money, this time keeping track of the individual

denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change.

§15.18. Equations

Forming totals is all very interesting in its way, but it's book-keeping rather than physics. As a glance at any school science textbook shows, the way to apply physics is to work out an unknown quantity - say, the time taken for a dropped ball to hit the ground - by combining known quantities into an equation - the height it is dropped from, and the strength of gravity.

It's a convention centuries old now that textbooks and research papers never describe these equations in running text. Even for simple formulae, we like to write "F=ma", not "let the force be the mass times the acceleration". And the standard way to print this is to break off and display an equation, not to squeeze it into the text as if it were ordinary verbiage. Just as Inform's Tables imitate those in printed books (see the next chapter), so its Equations do.

In this section, we'll use a combination of three equations to work out how soon and how hard an object pushed off a table will hit the floor. First, we'll include Metric Units, to define all of the kinds of value and notations we need.

```
Include Metric Units by Graham Nelson.
```

Now we'll give everything a mass (Metric Units likes to talk about mass instead of weight, but on Earth it's the same thing) and also set up a typical strength for gravity - it's a little less at the poles, a little more at the equator, but this is the conventional approximate value to use.

```
The acceleration due to gravity is an acceleration that varies. The
acceleration due to gravity is usually 9.807 m/ss. A thing has a mass.
The mass of a thing is usually 10g.
```

To a Renaissance scientist, typically living in a walled European town, a cannon ball was a familiar thing, and it often featured in imaginary experiments:

```
Laboratory is a room. The cannon ball is in the Laboratory. "A cannon
ball perches delicately on a lab bench." The mass of the cannon ball
```

```
is 2kg.
```

And now we're ready for the three equations. These will all have names, but we could just as easily have numbered them, calling them (say) "Equation 1", "Equation 2" and "Equation 3".

```
Equation - Newton's Second Law
    F=ma
where F is a force, m is a mass, a is an acceleration.
```

```
Equation - Principle of Conservation of Energy
    mgh = mv^2/2
where m is a mass, h is a length, v is a velocity, and g is the
acceleration due to gravity.
```

```
Equation - Galilean Equation for a Falling Body
    v = gt
where g is the acceleration due to gravity, v is a velocity, and t is an
elapsed time.
```

An equation has to take the form of one formula equals another, where each formula is made up from symbols defined afterwards. The symbols can be defined as definite values (as "g" is defined in the Galilean Equation), or just by telling Inform their kinds of value (as "v" and "t" are defined).

Equations are read using standard mathematical conventions. So "x + yz" means that we multiply y and z, then add that to x; "ab/cd" divides the product of a and b by the product of c and d. Multiplication signs can be omitted, just as science books normally do (though we can always write them if we want to, using the asterisk *, as usual in computing). The need for brackets is minimised, with any luck, but we can use them if we need to: "x(y+ab)" is legal, for instance.

One difference between Inform's conventions and mathematical ones, though, is that Inform generally ignores upper-versus-lower-case when reading variable names, so it wouldn't be a good idea to write "F = gMm/r^2" and expect "M" and "m" to be different from each other.

Here is the calculation:

```
Instead of pushing the cannon ball:
    let the falling body be the cannon ball;
```

```
    let m be the mass of the falling body;
    let h be 1.2m;
    let F be given by Newton's Second Law where a is the acceleration
due to gravity;
    let v be given by the Principle of Conservation of Energy;
    let t be given by the Galilean Equation for a Falling Body;
    say "You push [the falling body] off the bench, at a height of [h],
and, subject to a downward force of [F], it falls. [t to the nearest
0.01s] later, this mass of [m] hits the floor at [v].";
    now the falling body is in the location.
```

And the result is:

```
You push the cannon ball off the bench, at a height of 1.2m, and,
subject to a downward force of 19.614N, it falls. 0.49s later, this mass
of 2.0kg hits the floor at 4.85147 m/s.
```

Not all that fast-moving - it's only about 10 mph, ten times slower than one fired by a Renaissance cannon - but half a second wouldn't give you long to get your foot out of the way.

How was that done? The crucial lines are the ones in the form "let X be given by E...", which is a new form of "let".

let (a name not so far used) **be given by** (equation name)

or:
let (a temporary named value) **be given by** (equation name)

This phrase creates a new temporary variable, starting it with the value found by solving the given equation. The variable lasts only for the present block of phrases, which certainly means that it lasts only for the current rule.
Example:

```
    let F be given by Newton's Second Law where a is the
    acceleration due to gravity;
```

There is also a more compact syntax, giving the equation explicitly:

```
    let KE be given by KE = mv^2/2 where KE is an energy;
```

When we solve with "let", then, all of the other symbols should either already have values (because they exist as "let" values already made) or else be specified in the line. For instance,

```
let F be given by Newton's Second Law where a is the acceleration
due to gravity;
```

is allowed because "F" is one of the symbols in "F = ma"; of the other two symbols, we have a "let" variable called "m" already - it's the mass of the cannon ball - and we declare exactly what "a" is.

The next calculation is more interesting:

```
let v be given by the Principle of Conservation of Energy;
```

Since the equation here is "mgh = mv^2/2", Inform has to do some algebra to work out "v" in terms of the other unknowns - it's the square root of 2gh, but we don't need to work that out. Inform can't always solve implicit equations - for instance, it can't deduce "m" from this equation - but it's correct on all the easy cases which occur in basic physics, and that enables us to write equations in their most natural form, which is easier to read and understand.

The advantage of setting out an equation formally is that it can be used in many places - we could use Newton's Second Law again for something quite different, for example. But it's a little cumbersome for something simple which we only need once, so this is neater:

```
let KE be given by KE = mv^2/2 where KE is an energy;
```

Here the equation is written out explicitly instead of being named, but otherwise everything works in the same way.

Equations can also contain many of our standard functions, which are written for this purpose with their standard mathematical abbreviations. For example:

```
let x be given by sin x = 1 where x is a real number;
```

works out x as pi divided by 4, which is to say, 90 degrees. The Phrasebook entries on the mathematical functions give their abbreviations, but here they all are as a list:

```
abs, root, ceiling, floor, int, log, exp, sin, cos, tan, arcsin, arccos,
arctan, sinh, cosh, tanh, arcsinh, arccosh, arctanh
```

As an example, here's the definition of arcsinh given in the Standard Rules:

```
To decide which real number is the hyperbolic arcsine of (R - a real
number):
    let x be given by x = log(R + root(R^2 + 1)) where x is a real
number;
    decide on x.
```

Start of Chapter 15: Numbers and Equations
Back to §15.17. Totals
Onward to §15.19. Arithmetic with units
Example 263: ★ **Widget Enterprises** Allowing the player to set a price for a widget on sale, then determining the resulting sales based on consumer demand, and the resulting profit and loss.

§15.19. Arithmetic with units

The example equations in the previous section carried out quite a lot of arithmetic, but they may have given the impression that Inform always allows arithmetic - which is not true.

This is actually a good thing, because it keeps us from error. For instance, Inform will not allow:

```
Equation - Newton's Totally Bogus Law
    F = m^2
where F is a force, m is a mass.
```

because whatever you get when you square a mass, you don't get a force - in the same way that a length times another length makes an area, not another length.

Physicists call this "dimensional analysis", and it often provides clues about which equations are right. Just after the Second World War, someone correctly worked out the explosive power of an atomic bomb without any classified information simply by guessing what values would appear in the formula, and then finding the simplest equation they could appear in.

In general, Inform will not allow numerical kinds of value to be multiplied or divided by each other (or square or cube rooted) unless we give it instructions that this would make sense.

Of course, there's plenty we can still do without any need for such instructions. For instance, going back to weight,

```
The Weighbridge is a room. "A sign declares that the maximum load
is [100kg multiplied by 3]."
```

...will produce the text "A sign declares that the maximum load is 300kg." Here Inform knows that it makes sense to multiply a weight by 3, and that the result will be a weight. Similarly, Inform allows us to add and subtract weights, and several different forms of division are allowed:

```
The blackboard is in the Weighbridge. "A blackboard propped against
one wall reads: '122 / 10 is [122 divided by 10] remainder [remainder
after dividing 122 by 10]; but 122kg / 10kg is [122kg divided by 10kg]
remainder [remainder after dividing 122kg by 10kg]; and 122kg / 10 is
[122kg divided by 10] remainder [remainder after dividing 122kg by
10].'"
```

When we visit the Weighbridge, we find:

```
A blackboard propped against one wall reads: "122 / 10 is 12
remainder 2; but 122kg / 10kg is 12 remainder 2kg; and 122kg / 10 is
12kg remainder 2kg."
```

Whereas we are not allowed to divide 122 by 10kg: that would make no sense, since 122 is a number and not made up of kilograms. Inform will produce a problem message if we try. Similarly, Inform won't normally allow us to multiply two weights together - but see the next section.

Start of Chapter 15: Numbers and Equations
Back to §15.18. Equations

Onward to §15.20. Multiplication of units

Example 264: ★ **Frozen Assets** A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping.

Example 265: ★★ **Money for Nothing** An OFFER price FOR command, allowing the player to bargain with a flexible seller.

Example 266: ★★★ **Lemonade** Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

Example 267: ★★★ **Savannah** Using the liquid implementation demonstrated in Lemonade for putting out fires.

---

§15.20. Multiplication of units

To recap, then, it is forbidden to multiply 122kg and 10kg, not because it could never make sense (a scientist might occasionally multiply two weights) but because the result is - what? Not a number, and not a weight any more. But we are allowed to tell Inform what the result ought to be, and once we have done so, the multiplication will be allowed:

```
A length is a kind of value. 10m specifies a length. An area is a kind
of value. 10 sq m specifies an area.
```

```
A length times a length specifies an area.
```

```
The balance platform is in the Weighbridge. "The balance platform is
10m by 8m, giving it an area of [10m multiplied by 8m]."
```

which will turn up as:

```
The balance platform is 10m by 8m, giving it an area of 80 sq m.
```

And having told Inform that lengths multiply to area, we could also divide an area by a length to get a length: no further instructions would be needed.

The built-in "Metric Units" extension includes all of the standard ways that physical quantities are multiplied, and a good way to see these is to try out one of the Metric Units examples and look at the Kinds index, which includes a table showing how all of this works.

Start of Chapter 15: Numbers and Equations

Back to §15.19. Arithmetic with units

Onward to Chapter 16: Tables: §16.1. Laying out tables

Example 268: ★ **Depth** Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

Example 269: ★★ **Fabrication** A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

Example 270: ★★ **The Speed of Thought** Describing scientifically-measured objects in units more familiar to the casual

audience.

Examples from Chapter 15: Numbers and Equations

Start of this chapter
Chapter 16: Tables
Indexes of the examples

253 **ExampleAlias** *RB*

A telephone with phone numbers of the standard American seven-digit length.

Seven-digit telephone numbers are too long for Inform to handle when compiling to the Z-Machine, but they will work under Glulx. To have this example succeed, make sure that you have selected the Glulx option in your settings menu.

```
"Alias"

A telephone is a kind of thing. Understand "phone" or
"telephone" as a telephone.

A phone number is a kind of value. 999-9999 specifies a
phone number.
```

Now we borrow some techniques from the Understanding chapter to set up dialing actions:

```
Understand "dial [phone number] on [telephone]" as
dialing it on. Understand "dial [phone number] on
[something]" as dialing it on.
```

```
Understand the commands "phone" or "telephone" or
"call" as "dial".


Understand "call [text]" or "phone [text]" or "dial [text]" or
"telephone [text]" as a mistake ("That's not a number
you know.").


Dialing it on is an action applying to one phone number
and one thing.


Report dialing it on:
    say "You dial [the phone number understood]."
```

This much is enough to let us dial telephone numbers and have Inform report that we've done so; it doesn't actually provide a telephone system such that we could reach and converse with other characters (but see the other telephone examples in the recipe book for more on how one might do that).

We'll set up a little political espionage scenario from which our player can make calls:

```
The Senator's Junior Suite is a room. "The Senator
appears, unfortunately, to have very precise habits: little
in the room has been moved from its usual place; the
trash can is empty; the bed has been remade[if the blue
paper is unexamined]. There may in fact be nothing to
find here[end if]."


The bed is an enterable scenery supporter in the Junior
Suite.


The player is wearing a housekeeping uniform and a
brunette wig. The player carries a telephone called a
Nokia.
```

Borrowing again from the chapter on Understanding, we might arrange things so that the player knows and can call a few standard numbers with such syntax as CALL HOME:

```
Understand "home" as 555-9200.
```

And what if we'd like to have the player learn some phone numbers during the game?

```
A thing can be examined or unexamined. Carry out
examining something: now the noun is examined.


Understand "Stephen" as 555-2513 when the blue
paper is examined.
```

This will understand CALL STEPHEN once the paper is examined; before that, the player will just get the "That's not a number you know" response that Inform uses for all attempts to call unknown names.

We'd better plant this paper for the player to find:

```
The blue paper is in the drawer. The description of the
blue paper is "It reads: 'Call Stephen - 555-2513'."


The drawer is part of the dresser. It is closed and
openable. The dresser is in The Senator's Junior Suite.
The lamp is on the dresser. The description of the
dresser is "The single drawer is [if the drawer is
open]open[otherwise]shut[end if]."


Test me with "dial 555-9999 / call home on the
telephone / phone the president / call stephen / open
drawer / read paper / call stephen / put phone in drawer
/ close drawer / call stephen".
```

The player character's height is selected randomly at the start of play.

As with ordinary numbers, we can choose random units when this is useful:

```
"rBGH"

The Pharmaceutical Testing Facility is a room. "A [if the
player is short]large [end if][if the player is tall]cramped
[end if]white space with sterile counters and a[if the
player is tall]n uncomfortable little[end if] stool. There is
also a mirror, behind which someone must be watching
you. But you can't see through to that."

A counter, a one-way mirror, and a stool are scenery in
the Facility. The stool is an enterable supporter. The
counter supports a plate.

Height is a kind of value. 5 feet 11 inches specifies a
height. 5'11 specifies a height. A person has a height.

Definition: a person is tall if its height is 6 feet 0 inches
or more.

Definition: a person is short if its height is 5 feet 4 inches
or less.

When play begins:
    now the height of the player is a random height
between 5 feet 2 inches and 6 feet 4 inches;
```

now the right hand status line is "[height of player]".

Instead of examining the player:
    say "You, Test Subject, are [height of the player] tall."

The growth pill is a kind of thing. A growth pill is always edible. The description is usually "It is leaf-green and has a reassuring logo of a curling vine on the side. Nothing to worry about, nothing at all." Two growth pills are on the plate.

After eating the growth pill:
    increase the height of the player by 0 feet 6 inches;
    say "Your spine does something frightening and painful, and you find yourself looking down on the room from a wholly new angle.";
    try looking.

Test me with "examine me / eat pill / examine me / eat pill / examine me".

ExampleLethal Concentration 1 *RB*

A poisonous gas that spreads from room to room, incapacitating or killing the player when it reaches sufficient levels.

"Lethal Concentration"

A concentration is a kind of value. 200.9ppm specifies concentration. 200.9 ppm specifies concentration.

```
A room has a concentration called current
concentration. A room has a concentration called former
concentration.


Probability inverse is a number that varies. [This is
expressed as an inverse of the actual probability of
diffusion from one room to another, to avoid error.]
Probability inverse is 20. [That is, any given molecule of
gas has a 5% chance of leaving by a given doorway at
any given minute. Probability inverse should never drop
below 10, the maximum number of exits from the room.]


Every turn:
    follow the diffusion rules.


The diffusion rules are a rulebook.


A diffusion rule (this is the gas movement rule):
    repeat with space running through rooms:
        let sum be 0.0 ppm;
        repeat with way running through directions:
            let second space be the room way from the
space;
            if second space is a room:
                let difference be the former concentration of
the second space minus the former concentration of the
space;
                increase sum by the difference;
        let sum be sum divided by probability inverse;
        now current concentration of the space is the
former concentration of the space plus the sum.
```

A technical note: it would be possible to write "repeat with space running through rooms... repeat with second space running through rooms adjacent to the space" instead, but in practice this loops through all the rooms * all the rooms again * all the directions (to determine adjacency). Phrasing the loop this way omits the second multiplier. For a map of 25 rooms, this means that the loop runs 25 times faster than it would otherwise, and of course for a larger map the effect would be even more dramatic.

```
A diffusion rule (this is the resetting concentration rule):
    repeat with space running through rooms:
        now the former concentration of the space is the
current concentration of the space.
```

```
The last diffusion rule (this is the lethal dosage rule):
    if the current concentration of the location is greater
than LC50:
        say "The concentration in the air overpowers
you...";
        end the story;
    otherwise:
        if the current concentration of the location is
greater than TLV-STEL:
            say "You feel extremely uncomfortable in this
environment."
```

```
Instead of doing something when the current
concentration of the location is greater than TLV-STEL:
    if going, continue the action;
    say "You can't work in this environment: your eyes
and nose sting and it hurts to breathe."
```

And, for testing purposes, a square grid of rooms:

```
Room 1A is west of Room 1B. Room 1B is west of
Room 1C. Room 1C is west of Room 1D. Room 1D is
west of Room 1E.
```

```
Room 2A is south of room 1A and west of Room 2B.
Room 2B is west of Room 2C and south of Room 1B.
Room 2C is west of Room 2D and south of Room 1C.
Room 2D is west of Room 2E and south of Room 1D.
Room 2E is south of Room 1E.
```

```
Room 3A is south of room 2A and west of Room 3B.
Room 3B is west of Room 3C and south of Room 2B.
```

```
Room 3C is west of Room 3D and south of Room 2C.
Room 3D is west of Room 3E and south of Room 2D.
Room 3E is south of Room 2E.
```

```
Room 4A is south of room 3A and west of Room 4B.
Room 4B is west of Room 4C and south of Room 3B.
Room 4C is west of Room 4D and south of Room 3C.
Room 4D is west of Room 4E and south of Room 3D.
Room 4E is south of Room 3E.
```

```
Room 5A is south of room 4A and west of Room 5B.
Room 5B is west of Room 5C and south of Room 4B.
Room 5C is west of Room 5D and south of Room 4C.
Room 5D is west of Room 5E and south of Room 4D.
Room 5E is south of Room 4E.
```

```
The former concentration of room 3C is 800.0 ppm.
```

For variety of testing, here is another room set-up, this time with some corridors and walls within; uncommenting it, and commenting out the connected grid, will let us explore what would happen in alternative cases, to prove to ourselves that the model works consistently.

```
[Room 1A is west of Room 1B. Room 1B is west of
Room 1C. Room 1C is west of Room 1D. Room 1D is
west of Room 1E.
```

```
Room 2A is west of Room 2B. Room 2B is west of
Room 2C. Room 2C is west of Room 2D. Room 2D is
west of Room 2E. Room 2E is south of Room 1E.
```

```
Room 3A is south of room 2A and west of Room 3B.
Room 3B is west of Room 3C. Room 3C is west of
Room 3D. Room 3D is west of Room 3E.
```

```
Room 4A is west of Room 4B. Room 4B is west of
Room 4C. Room 4C is west of Room 4D. Room 4D is
west of Room 4E. Room 4E is south of Room 3E.


Room 5A is south of room 4A and west of Room 5B.
Room 5B is west of Room 5C and south of Room 4B.
Room 5C is west of Room 5D and south of Room 4C.
Room 5D is west of Room 5E and south of Room 4D.
Room 5E is south of Room 4E.]
```

For the sake at least of seeing what's going on in the example, let's also provide the player with the means to view the gas diffusion graphically:

```
The status grid is a device carried by the player. The
status grid is switched on.


Every turn:
    try examining the grid.


Instead of examining the status grid:
    say "[fixed letter spacing][bar][line break]";
    say "|[state of room 1A]|[state of room 1B]|[state of
room 1C]|[state of room 1D]|[state of room 1E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 2A]|[state of room 2B]|[state of
room 2C]|[state of room 2D]|[state of room 2E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 3A]|[state of room 3B]|[state of
room 3C]|[state of room 3D]|[state of room 3E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 4A]|[state of room 4B]|[state of
room 4C]|[state of room 4D]|[state of room 4E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 5A]|[state of room 5B]|[state of
room 5C]|[state of room 5D]|[state of room 5E]|[line
break]";
    say "[bar][variable letter spacing][line break]".
```

```
To say bar:
    say "-----------------------------------------------".


TLV is a concentration that varies. TLV is 30.0ppm.
[Long-term exposure maximum, safe for 8 hours a day.]


TLV-STEL is a concentration that varies. TLV-STEL is
50.0ppm. [Short-term exposure maximum, safe for
fifteen minutes max.]


TLV-C is a concentration that varies. TLV-C is
150.0ppm. [Absolute exposure ceiling.]


LC50 is a concentration that varies. LC50 is 300.0ppm.
[Concentration at which 50 percent of test subjects die
of exposure, usually expressed in terms of time and
body weight; in our LC50 these are factored in for the
player's weight for one minute.]
```

The values set for these would depend on the type of poisonous gas in question; we'd want to adjust appropriately.

```
To say state of (space - a room):
    if the space is the location, say bold type;
    if current concentration of space is less than
10.0ppm, say " ";
    if current concentration of space is less than
100.0ppm, say " ";
    say current concentration of space;
    say roman type.
```

Now, in theory we might also want to account for sources and sinks, items that either inject poisonous gas into the environment or remove it again. For simplicity, we will assume that these contributions can also be calculated in ppm and that the total number of inert and poisonous gas molecules in a room never changes (so if poison gas molecules are

added, an equal number of inert molecules are removed). If room pressure were able to change, our model would have to be improved, so let us assume for now that that never happens. We want this sink/source business to calculate before any other portion of the diffusion rulebook, so set it as a first diffusion rule.

```
A gas source is a kind of thing. A gas source has a
concentration called the contribution. The contribution of
a gas source is usually 30.0ppm.


Room 2B contains a gas source called a spigot. The
contribution of the spigot is 50.0ppm. Room 5A contains
a gas source.


A gas sink is a kind of thing. A gas sink has a
concentration called the contribution. The contribution of
a gas sink is usually 30.0ppm.


Room 5E contains a gas sink called a fan. The
contribution of the fan is 80.0ppm.


The first diffusion rule (this is the sources and sinks
rule):
    follow the sources rule;
    follow the sinks rule.


This is the sinks rule:
    repeat with item running through gas sinks:
        let space be the location of the item;
        decrease the former concentration of the space by
the contribution of the item;
        if the former concentration of the space is less than
0.0ppm, now the former concentration of the space is
0.0ppm.
```

```
This is the sources rule:
    repeat with second item running through gas sources:
        let space be the location of the second item;
        increase the former concentration of the space by
the contribution of the second item;
        if the former concentration of the space is less than
0.0ppm, now the former concentration of the space is
0.0ppm.



Test me with "z / z / z / z / z / z / z / z".
```

Hiking Mount Rainier, with attention to which locations are higher and which lower than the present location.

Suppose we have a landscape with a great deal of up and down variation, where GO UP and GO DOWN will be significant almost everywhere, and specifying them all individually a tremendous pain:

```
"Wonderland"

An altitude is a kind of value. 1000 feet specifies an
altitude. A room has an altitude.

Definition: a room is low if its altitude is 3000 feet or
less. Definition: a room is high if its altitude is 5000 feet
or more.

Instead of going down:
    if an adjacent room is lower than the location:
        let the valley be the lowest adjacent room;
        let the way be the best route from the location to
the valley;
        say "(that is, [way])[paragraph break]";
```

```
        try going the way;
    otherwise:
        say "You're in a local valley: there's no down from
here."

Instead of going up:
    if an adjacent room is higher than the location:
        let the peak be the highest adjacent room;
        let the way be the best route from the location to
the peak;
        say "(that is, [way])[paragraph break]";
        try going the way;
    otherwise:
        say "You're on a local peak."
```

Paradise is a room. Paradise has altitude 5400 feet. "A handsome parking lot, a picnic ground, and the Henry M. Jackson Memorial Visitor Center. The latter offers, for serious climbers, a hot shower; for nature enthusiasts, an interpretive museum; and for car-trippers, a gift shop selling canned slugs. All of which is a largely unsuccessful distraction from the peak of Mt. Rainier beyond."

Cougar Rock is southwest of Paradise. The altitude of Cougar Rock is 3180 feet. "Numerous individual campsites and (on the road inventively labeled 'F') a handful of larger campgrounds suitable for church groups and family reunions."

Longmire is southwest of Cougar Rock. It has altitude 2760 feet. "A tiny town: it has to offer a few groceries, a post office, and a lodge for people who do not care to camp, all built in a rustic Park Service way."

Panorama Point is north of Paradise. It has altitude 6800 feet. Camp Muir is north of Panorama Point. It has altitude 10188 feet. Columbia Crest is northwest of Camp Muir. It has altitude 14410 feet. St Andrews Rock

```
is west of Columbia Crest. It has altitude 10992 feet.
Camp Schuman is northeast of Columbia Crest. It has
altitude 9510 feet.
```

Since Mount Rainier National Park runs to over 235,000 acres, we will omit the rest of the locations, but it does seem fair to give a little more credit to anyone who makes the summit:

```
Instead of going up in the highest room:
    say "You're standing at the summit of Mt. Rainier, the
highest point in the state of Washington. There is no
up."


Test me with "up / up / up / down / down / up / up".
```

---

257 **Example** **Lethal Concentration 2** *RB*

Poisonous gas again, only this time it sinks.

This is a slight variation on the previous gas diffusion example: the main difference is that gas preferentially moves towards lower rooms, and will gradually settle in the bottom floor. We do this by calculating the probability of movement separately for each pair of rooms.

```
"Lethal Concentration"


A concentration is a kind of value. 200.9ppm specifies
concentration. 200.9 ppm specifies concentration.


A room has a concentration called current
concentration. A room has a concentration called former
concentration.
```

```
To decide what number is the probability inverse
between (space - a room) and (second space - a room):
    let guess be 20;
    let way be the best route from space to second
space;
    if way is up, let guess be 50;
    if way is down, let guess be 10;
    if the guess is less than 10, decide on 10;
    decide on guess.
```

If we wanted, we could introduce other concerns into the calculation here: open and closed doors, windows between rooms, rooms that are outdoors vs. those that are indoors, and so on. The possibilities are numerous, so we will stick with the simple principle that our poison gas sinks.

```
Every turn:
    follow the diffusion rules.
```

```
The diffusion rules are a rulebook.
```

```
A diffusion rule (this is the gas movement rule):
    repeat with space running through rooms:
        let sum be 0.0 ppm;
        repeat with way running through directions:
            let second space be the room way from the
space;
            if second space is a room:
                let incoming be the former concentration of
the second space divided by the probability inverse
between second space and space;
                let outgoing be the former concentration of the
space divided by the probability inverse between space
and second space;
                let difference be incoming minus outgoing;
                increase sum by the difference;
        now current concentration of the space is the
former concentration of the space plus the sum.
```

```
A diffusion rule (this is the resetting concentration rule):
    repeat with space running through rooms:
```

now the former concentration of the space is the current concentration of the space.

The last diffusion rule (this is the lethal dosage rule):
    if the current concentration of the location is greater than LC50:
        say "The concentration in the air overpowers you...";
        end the story;
    otherwise:
        if the current concentration of the location is greater than TLV-STEL:
            say "You feel extremely uncomfortable in this environment."

Instead of doing something when the current concentration of the location is greater than TLV-STEL:
    if going, continue the action;
    say "You can't work in this environment: your eyes and nose sting and it hurts to breathe."

Room 1A is west of Room 1B. Room 1B is west of Room 1C. Room 1C is west of Room 1D. Room 1D is west of Room 1E.

Room 2A is west of Room 2B and below room 1A.
Room 2B is west of Room 2C and below Room 1B.
Room 2C is west of Room 2D and below Room 1C.
Room 2D is west of Room 2E and below Room 1D.
Room 2E is south of Room 1E and below Room 1E.

The former concentration of Room 1C is 800.0 ppm.

The status grid is a device carried by the player. The status grid is switched on.

And just for fun, this time we'll make the grid prettier, too; but this will work only on the Z-machine setting, not Glulx.

```
Every turn:
    try examining the grid.



    Instead of examining the status grid:
    say "[unicode box drawings light down and right][top
bar][unicode box drawings light down and left][line
break]";
    say "[unicode box drawings light vertical]";
    say "[state of room 1A][state of room 1B][state of
room 1C][state of room 1D][state of room 1E]
upstairs[line break]";
    say "[unicode box drawings light vertical and right]
[middle bar][unicode box drawings light vertical and left]
[line break]";
    say "[unicode box drawings light vertical]";
    say "[state of room 2A][state of room 2B][state of
room 2C][state of room 2D][state of room 2E]
downstairs[line break]";
    say "[unicode box drawings light up and right][bottom
bar][unicode box drawings light up and left][variable
letter spacing][line break]"



Include Unicode Character Names by Graham Nelson.



To say top bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say
"[unicode box drawings light down and horizontal]";
        otherwise say "[unicode box drawings light
horizontal]".



To say middle bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say
"[unicode box drawings light vertical and horizontal]";
        otherwise say "[unicode box drawings light triple
```

dash horizontal]".

To say bottom bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say "[unicode box drawings light up and horizontal]";
        otherwise say "[unicode box drawings light horizontal]".

TLV is a concentration that varies. TLV is 30.0ppm. [Long-term exposure maximum, safe for 8 hours a day.]

TLV-STEL is a concentration that varies. TLV-STEL is 50.0ppm. [Short-term exposure maximum, safe for fifteen minutes max.]

TLV-C is a concentration that varies. TLV-C is 150.0ppm. [Absolute exposure ceiling.]

LC50 is a concentration that varies. LC50 is 300.0ppm. [Concentration at which 50 percent of test subjects die of exposure, usually expressed in terms of time and body weight; in our LC50 these are factored in for the player's weight for one minute.]

Include Basic Screen Effects by Emily Short.

To say state of (space - a room):
    if the current concentration of space is less than TLV, say blue letters;
    if the current concentration of space is TLV, say blue letters;
    if the current concentration of space is greater than TLV, say green letters;

```
    if the current concentration of space is greater than
TLV-STEL, say yellow letters;
    if the current concentration of space is greater than
TLV-C, say red letters;
    say "[unicode square with diagonal crosshatch fill]";
    say default letters;
    say "[unicode box drawings light vertical]".



Test me with "z / z / z / z / z / z / z / z".
```

ExampleDimensions *RB*

258

This example draws together the previous snippets into a working implementation of the weighbridge.

The following is not a very sophisticated approach, because it does not allow for weight to accumulate: if we put a gold ingot into a paper bag, then put the bag on the balance platform, only the bag's weight will register. But it will do for a first try.

```
"Dimensions"



A length is a kind of value. 10m specifies a length. An
area is a kind of value. 10 sq m specifies an area. A
length times a length specifies an area.



A weight is a kind of value. 10kg specifies a weight.
Everything has a weight.



The verb to weigh means the weight property. A thing
usually weighs 1kg.
```

Definition: A thing is light if its weight is 3kg or less.

Definition: A thing is heavy if its weight is 10kg or more.

The Weighbridge is a room.

A blackboard is in the Weighbridge. "A blackboard propped against one wall reads: '122/10 is [122 divided by 10] remainder [remainder after dividing 122 by 10]; 122kg/10kg is [122kg divided by 10kg] remainder [remainder after dividing 122kg by 10kg]; 122kg/10 is [122kg divided by 10] remainder [remainder after dividing 122kg by 10].'" The blackboard weighs 10kg.

A feather and a lead pig are in the Weighbridge. The lead pig weighs 45kg.

The balance platform is a supporter in the Weighbridge. "The balance platform is 10m by 8m, giving it an area of [10m multiplied by 8m], currently weighing [the list of things on the platform]. The scale alongside reads: [total weight of things on the platform]. [if two things are on the platform]Average weight is: [the total weight of things on the platform divided by the number of things on the platform]. Heaviest item is [the heaviest thing on the platform], at [weight of the heaviest thing on the platform]. Lightest item is [the lightest thing on the platform], at [weight of the lightest thing on the platform]. [otherwise]It seems to be able to weigh several things at once."

Test me with "get feather / put it on platform / look / get pig / put it on platform / look".

To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster.

The following shows the kind of "realism" rules which could be introduced using weights. Not entirely realistic: we do not bother to rupture containers out of the player's sight.

```
"Lead Cuts Paper"

A weight is a kind of value. 10kg specifies a weight.
Everything has a weight. A thing usually has weight 1kg.

A container has a weight called breaking strain. The
breaking strain of a container is usually 50kg. Definition:
A container is bursting if the total weight of things in it is
greater than its breaking strain.

A lead pig, a feather, a silver coin and a paper bag are
in a room called the Metallurgy Workshop. The paper
bag is a container with breaking strain 2kg. The lead pig
has weight 50kg.

Every turn when a container (called the sack) held by
someone visible (called the chump) is bursting:
    say "[The sack] splits and breaks under the weight! [if
the player is the chump]You discard[otherwise][The
chump] discards[end if] its ruined remains, looking
miserably down at [the list of things in the sack] on the
floor.";
    now all of the things in the sack are in the location;
    now the sack is nowhere.

Test me with "get bag / get feather / put feather in bag /
```

get pig / put pig in bag / look".

---

260 **ExampleZqlran Era 8** *RB*

Creating an alternative system of time for our game, using new units.

Suppose that our game takes place on an alien planet that does not follow Earth time. On this planet, we want to track time with different units. We also want time to advance in those units, and we want to be able to set a schedule of timed events.

```
"Zqlran Era 8"

The Barren Lavender Surface of Zql is a room. "It is late
twilight on Zql. Overhead, two crescent moons, both
green, mark the sluggish passage of time. A cold wind is
blowing over the pale purplish ground cover, but it does
not penetrate your airtight suit."

A Zqlran date is a kind of value. 14-88 specifies a Zqlran
date with parts zqls and frbs. Current zqlran date is a
zqlran date that varies. The current zqlran date is 8-22.
Previous zqlran date is a zqlran date that varies. The
previous zqlran date is 8-20.

When play begins:
    now left hand status line is "[current zqlran date], or
[current zqlran date in words]".

To say (Zqlra - a Zqlran date) in words:
    say "[zqls part of Zqlra] Z, [frbs part of Zqlra] f."
```

Inform automatically supplies a way to say a new unit, which will look similar to the format in which we defined that unit in the first place. But

we can (as shown here) create our own alternative say phrases to express the units in other ways as well.

Next, we need to meddle with time advancement so that time is tracked in Zqlran date rather than in minutes. This requires borrowing a trick from a later chapter, to replace Inform's built-in time handling with an alternative time handling rule of our own:

```
The Zqlran time rule is listed instead of the advance
time rule in the turn sequence rules.


This is the Zqlran time rule:
    increment turn count;
    now the previous zqlran date is current zqlran date;
    increase the current zqlran date by 0-02;
    repeat through the Table of Zql Schedule:
        if era entry is greater than previous zqlran date and
era entry is not greater than current zqlran date:
            say event entry;
            say paragraph break;
            blank out the whole row.


Table of Zql Schedule

era                event
8A24               "A wisp-thin cloud blows rapidly across the face of Nepenthe, the
lesser of the two green moons."
8T28               "The cloud across Nepenthe clears."
```

Note that we could if we wished use a different device for scheduling events: this one simply prints text at scheduled eras, but we might also (for instance) make the event entry be a rule for Inform to follow, and tell Inform to carry out that rule at the scheduled time.

ExampleSnip

A string which can be cut into arbitrary lengths, and then tied back together.

RB

"Snip!"

Length is a kind of value. 30 inch specifies a length. 20 in specifies a length. 50 inches specifies a length.

A string is a kind of thing. A string has a length. The length of a string is usually 36 inches.

Before printing the name of a string, say "[length] piece of ". Rule for printing the plural name of a string: say "[length] pieces of string".

Understand the command "cut" as something new. Understand "cut [length] from/off [something]" as trimming it by (with nouns reversed). Understand "cut [something] by [length]" as trimming it by. Understand the command "trim" as "cut".

Trimming it by is an action applying to one thing and one length.

Check trimming it by:
    if the length understood is 0 inches, say "You're approaching Zeno's string at this point." instead;
    if the length understood is greater than the length of the noun, say "[The noun] is only [length of the noun] long to start with." instead;
    if the length understood is the length of the noun, say "[The noun] is already exactly [length of the noun] long." instead.

Carry out trimming it by:

```
    now the length of the noun is the length of the noun
minus the length understood;
    let the other half be a random string in the string
repository;
    now the length of the other half is the length
understood;
    move the other half to the player.


Report trimming it by:
    reset string lengths; [we will define this in a moment;
it helps guarantee that our descriptions of the strings are
correct when we write the output list]
    say "You now have [a list of strings carried by the
player]."


Understand "cut [something] in half" as halving. Halving
is an action applying to one thing.


Carry out halving:
    let half measure be the length of the noun divided by
2;
    now the length understood is half measure;
    try trimming the noun by half measure.
```

This fudges slightly, since an odd-length string will be divided into uneven halves. Keeping track of fractional inches would complicate matters, though, so let's assume for now that this doesn't matter.

```
The player carries a string.


The Scissors Room is a room.


The string repository contains 35 strings.
```

Since our initial string is 36 inches long and it is impossible for the player to divide it into pieces smaller than an inch each, we need a total of 36

items to represent all the string-bits: one that the player carries at the outset, and 35 others. We should bear in mind that it is usually a good idea to use the smallest number of spare objects we can get away with: writing a game that required 1000 strings in the string repository would place silly demands on the resources of the system, so it's best to avoid that sort of thing if possible.

Now with a bit of fiddling we can also teach Inform to recognize descriptors such as "the shortest string":

```
Ordinariness is a kind of value. The ordinarinesses are
longest, medium, shortest. A string has an ordinariness.
Understand the ordinariness property as referring to a
string.


Definition: a string is small if its length is 2 in or less.
Definition: a string is large if its length is 20 in or more.


Before reading a command:
    reset string lengths.


To reset string lengths:
    let upper measure be the length of the largest visible
string;
    let lower measure be the length of the smallest visible
string;
    repeat with item running through strings:
        now the ordinariness of the item is medium;
        if the length of the item is the upper measure, now
the item is longest;
        if the length of the item is the lower measure, now
the item is shortest.


After reading a command:
    if the player's command includes "shorter", replace
the matched text with "shortest";
    if the player's command includes "longer", replace the
matched text with "longest".
```

```
Instead of tying a string to a string:
    move the second noun to the string repository;
    now the length of the noun is the length of the noun
plus the length of the second noun;
    decrease the length of the noun by 1 inch;
    say "You end up with [a noun], as some is taken up
by the knot."
```

This is still a little incomplete because we cannot refer to strings by their lengths, as in "the 2 inch string" and so on. To do this, we borrow a line from the chapter on Understanding:

```
Understand the length property as referring to a string.
```

```
Test me with "trim string by 4 in / cut longer string in half
/ cut longest string in half / cut shortest string in half / g /
g / tie longest string to shortest string / tie longest string
to medium string / i / x 16 inch string / drop 8 inch string
/ i".
```

---

262 **ExampleNickel and Dimed** *RB*

A more intricate system of money, this time keeping track of the individual denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change.

Typically games which keep track of the player's wealth need only do so as an abstract number, but occasionally it becomes useful to represent money as physical coins and bills. Here is an example that does exactly that:

```
"Nickel and Dimed"
```

```
Section 1 - Currency
```

```
Price is a kind of value. $10.99 specifies a price with
parts dollars and cents. A thing has a price. The price of
a thing is usually $0.00.


Money is a kind of thing. Coin is a kind of money.


A dollar bill is a kind of money. The price of a dollar bill
is $1.00. The printed name of a dollar bill is "dollar bill".
Rule for printing the plural name of a dollar bill: say
"dollar bills". The description of a dollar bill is "It has
George Washington's head on the front, which with a bit
of creative folding can be scrunched to look like a
mushroom. All important things really are learned in
kindergarten."


A five-dollar bill is a kind of money. The price of a five-
dollar bill is $5.00. Understand "five" or "five dollar" as
the five-dollar bill. The description of a five-dollar bill is
"Abraham Lincoln. He looks slightly less dignified here
than he does on the penny."


A hundred-dollar bill is a kind of money. The price of the
hundred-dollar bill is $100.00. Understand "hundred" or
"hundred dollar" as the hundred-dollar bill. Understand
"dollar" as the dollar bill. The description of a hundred-
dollar bill is "It's got Benjamin Franklin, who always gets
shafted: a denomination too large for anyone to carry
conveniently, and a lot of local fame in Philadelphia."
```

Our choice of understand rules guarantees that "five dollar" will be understood as the five, but "dollar" alone as the single. We will learn more about "understand" in later chapters, but here is a test to check the functionality:

```
Test bills with "x hundred dollar bill / x five dollar bill / x
hundred / x five / x dollar / x dollar bill".
```

A quarter is a kind of coin. The price of a quarter is $0.25. The description of a quarter is "One of the old-fashioned variety, not a state quarter."

A dime is a kind of coin. The price of a dime is $0.10. The description of a dime is "Franklin Roosevelt, trying not to look too annoyed that his coin is so small and thin."

A nickel is a kind of coin. The price of a nickel is $0.05. The description of a nickel is "A chubby coin, but you've always liked Thomas Jefferson, and the Monticello on the back is a nice touch."

A penny is a kind of coin. The price of a penny is $0.01. The description of the penny is "A profile of Abe Lincoln. Sometime soon they'll stop minting these, you're sure of it."

Section 2 - Ownership

Ownership relates one person (called the owner) to various things. The verb to own means the ownership relation.

Definition: a thing is owned if the player owns it.

Instead of buying something which is owned by the player:
    say "You already own [the noun]."

Instead of going somewhere when the player encloses something (called the stolen goods) which is not owned by the player:
    if the owner of the stolen goods is not a person:
        now the player owns the stolen goods;
        continue the action;
    if the owner of the stolen goods can see the player,
        say "'Hey there buddy, not so fast,' says [the owner of the stolen goods]. 'You going to buy [the stolen goods] first, or am I gonna call the cops?'";
    otherwise continue the action.

After taking inventory:
    say "Altogether, you've got [the player's cash] on your person."

To decide what price is the player's cash:
    let sum be the total price of money enclosed by the player;
    decide on sum.

To decide what price is the sum in (item - a container):
    let sum be the total price of the money in the item;
    decide on sum.

When play begins: now every thing carried by the player is owned by the player.

Section 3 - Purchasing and Sales

Definition: a thing is worthless if the price of it is $0.00.
Definition: a thing is valuable if it is not worthless.

```
A thing can be for sale.


Rule for printing room description details of something
(called target) which is for sale (this is the disclose
prices in room description rule): say " ([price of the
target])".


Before listing contents: group money together giving
articles.


Instead of examining a for sale thing (this is the describe
things by price rule):
    say "[The noun] costs [the price of the noun], payable
to [the owner of the noun]."
```

The cashbox is a theoretical construct, not something the player will ever encounter in the course of the game. It contains all the money that is available for non-player characters to use in making change. If we wanted, we could give each character his own stash of change, but this would increase the likelihood that any given person would run out of cash to make change with. (And in this example there is only one vendor anyway.)

```
The cashbox is a container. The cashbox contains 10
pennies. The cashbox contains 10 nickels. The cashbox
contains 10 dimes. The cashbox contains 10 quarters.
The cashbox contains 10 dollar bills. The cashbox
contains 10 five-dollar bills.


The block buying rule is not listed in the check buying
rules.


Check buying something:
    if the noun is not for sale, say "[The owner of the
noun] does not want to sell you [the noun]." instead;
    if the player's cash is less than the price of the noun,
```

```
say "You can't afford the asking price of [the price of the
noun] for [the noun]." instead.



Carry out buying something:
    let sum paid be $0.00;
    while sum paid is less than the price of the noun:
        let current target be the price of the noun minus the
sum paid;
        let bill offered be the best money from the player
for the current target;
        if the bill offered is money:
            move the bill offered to the owner of the noun;
            now the bill offered is spent;
            increase the sum paid by the price of the bill
offered;
            let current target be the price of the noun minus
the sum paid;
    say "You hand [the owner of the noun] [a list of spent
money]. [run paragraph on]";
    let change be $0.00;
    if the sum paid is greater than the price of the noun:
        now the change is the sum paid minus the price of
the noun;
    if change is greater than the sum in the cashbox:
        now the player carries every spent money;
        now every spent thing is fresh;
        say "'Whoa,' says [the owner of the noun], handing
the cash back to you. 'I can't make change for that,
man, sorry.'" instead;
    now every spent thing is in the cashbox;
    now every spent thing is fresh;
    while change is greater than $0.00:
        let change bill be the best money from the cashbox
for change;
        decrease change by the price of the change bill;
        now change bill is spent;
        move change bill to player;
    if money is spent, say "[The owner of the noun]
makes change with [a list of spent money]. [run
paragraph on]";
    now every spent thing is fresh;
    if the noun is not enclosed by the player and the
owner of the noun can touch the noun:
        say "'Here ya go,' says [the owner of the noun],
handing [the noun] to you. [run paragraph on]";
        move the noun to the player;
    now the player owns the noun.
```

```
Money can be spent or fresh.


Report buying something:
    if the player owns the noun,
        say "Your transaction is now complete, leaving you
with [the player's cash]."
```

We've skipped over defining what makes a denomination the best for a given transaction, so we'd better do that now. Our goal is to avoid ever having the player gratuitously overpay -- he should always offer the smallest amount of money that will meet the price of what he's buying.

We also assume that all money "enclosed by the buyer" -- that is, somewhere in the buyer's possession -- is available for use. This might not be true in a game where the player could pick up, say, a sealed lucite container with a ten-dollar bill inside; in that case we would have to define our terms more rigorously, perhaps by requiring that the bills be both enclosed and touchable by the buyer. The touchability check adds an extra layer of calculation, however, and since it is not necessary in this example (and probably not in most other cases either), we'll leave it out:

```
Definition: money is costly if its price is $2.50 or more.
Definition: money is cheap if its price is $0.99 or less.


Functional relation is a kind of value. The functional
relations are overpayment, underpayment and
irrelevant. Money has a functional relation.


To decide what money is the best money from (buyer -
a thing) for (cost - a price):
    repeat with bill offered running through money:
        if the bill offered is enclosed by the buyer:
            if the price of the bill offered is the cost, decide
on the bill offered;
            if the price of the bill offered is greater than the
cost, now the functional relation of bill offered is
overpayment;
```

```
            otherwise now the functional relation of the bill
        offered is underpayment;
        otherwise:
            now the functional relation of the bill offered is
        irrelevant;
    [say "underpayment: [a list of underpayment money]
    overpayment: [a list of overpayment money]";]
    if the total price of underpayment money is less than
        the cost:
        decide on the cheapest money which is
        overpayment;
    otherwise:
        decide on the costliest money which is
        underpayment.
```

Notice the "say underpayment/overpayment section..." noted out, above. This is for debugging purposes: when writing complex code, it is sometimes useful to put in lines that will say explicitly what is going on. We can enclose them in brackets and Inform will ignore them as though they were comments; if we run into any problems with the code later, we can erase the brackets and see the diagnostic printed to the screen as we play.

```
Instead of giving money to someone:
    say "Best to keep the transaction simple by buying
    whatever you want."
```

```
Section 4 - The Scenario
```

```
The player carries 2 dollar bills. The player carries a
nickel. The player carries 2 pennies. The player carries
a five-dollar bill. The player carries 1 hundred-dollar bill.
```

```
The Subway Station is a room.
```

```
The Bitterly Cold Street is north of the Subway Station.
"Even though there is no actual snow or ice, the street is
about as cold as you can stand, for which reason
walking the twenty blocks uptown is not an acceptable
```

```
option." The Bitterly Cold Street contains a dollar bill.


The newspaper man is a man in the Subway Station. "A
newspaper man in a knit cap and fingerless gloves is
hopping up and down behind his stand[if the turn count
is 1]. Cold weather, caffeine overdose, or mental
illness? You may never know. Welcome to New
York[end if]." The description is "Eye contact with
strangers is something to avoid around here."


The stand is a supporter in the Station. The stand is
scenery.


A copy of the New York Times is on the stand. The price
of the New York Times is $1.25.


A pack of gum is on the stand. The price of the gum is
$0.40.


A paperback novel is on the stand. The price of the
paperback novel is $7.99.


A packet of trading cards is on the stand. The price of
the packet is $0.99.


When play begins:
    now every thing on the stand is owned by the
newspaper man;
    now every thing on the stand is for sale.
```

We could have done all that by hand, but the initialization requires a little less work.

The ticket machine is a container in the Subway Station. It is fixed in place. The description of the ticket machine is "An LED screen on the front instructs you to insert [remaining ticket total] to complete your purchase. You also notice that the NO CHANGE light is lit up." The light is part of the ticket machine. The printed name of the light is "no change light". Understand "no change" or "no change light" as the light.

The description of the light is "In the whole of your recollection, the ticket machine has actually had change a total of twice. Usually, as now, the no-change light gleams angrily, daring you to put in more than you owe." A cash return button is part of the ticket machine. Instead of pushing the cash return button: say "The ticket machine regurgitates [the list of things in the ticket machine]."; now every thing in the ticket machine is carried by the player. Instead of taking something which is in the ticket machine: say "The ticket machine has swallowed your money, but it can be retrieved (you hope) with the cash return button."

Instead of inserting a hundred-dollar bill into the ticket machine:
    say "What, are you nuts?"

To decide what price is the remaining ticket total:
    let absolute cost be $2.25;
    let remaining cost be absolute cost minus the total price of things in the ticket machine;
    if remaining cost is less than $0.00, decide on $0.00;
    decide on remaining cost.

Instead of inserting something which is not money into the ticket machine: say "The ticket machine only accepts money, not other tokens of your esteem and regard."

Instead of inserting a penny into the ticket machine:

```
    say "The penny rattles out again mockingly: not even
the ticket machine thinks these are worth anything."
```

```
A subway pass is a kind of thing. 15 subway passes are
in the cashbox. The description of a subway pass is
usually "A rectangle of thick lavender paper with a black
magnetic stripe running up the back side. It is good for
one trip on the subway."
```

```
After inserting something into the ticket machine:
    if the remaining ticket total is $0.00:
        let purchased ticket be a random subway pass in
the cashbox;
        if purchased ticket is not a subway pass, say "The
ticket machine grunts disobligingly and then the
unwelcome word MALFUNCTION parades across the
LED screen, three letters at a time." instead;
        repeat with item running through things in the
machine:
            now the item is nowhere;
        move purchased ticket to player;
        say "The ticket machine beeps obligingly and
disgorges a single subway pass.";
    otherwise:
        say "The ticket machine beeps obligingly and
adjusts its price down to [remaining ticket total]."
```

And because even though the ticket machine is a container, we don't want to say (empty) after it in the room description:

```
Rule for printing room description details of the ticket
machine:
    do nothing instead.
```

```
Test me with "buy novel / n / get dollar / s / buy novel".
```

After all that, we should probably give the player a chance to win, as well:

```
The turnstile is south of the Subway Station. "A turnstile
```

is all that separates you from the subway platform stairs." The turnstile is north of the Platform. The turnstile is a door. Before going down in the presence of the turnstile, try going south instead. The turnstile is openable. The turnstile is open.

Instead of going through the turnstile when the player carries a subway pass: say "You enter the turnstile and begin your journey uptown..."; end the story finally saying "At last". Instead of going through the turnstile: say "You can't go through the turnstile without a subway pass. They're very strict about this."

Instead of inserting money into the turnstile: say "The turnstile takes passes, not money." Instead of inserting a subway pass into the turnstile, try entering the turnstile.

Test more with "buy times / put all but five-dollar bill in machine / press button / buy gum / buy cards / i / put dollar in machine / g / put quarter in machine / i / d".

In fairness to the Metropolitan Transit Authority, we should admit that most of the ticketing machines in the real New York subway are better than this, and will accept, say, a debit card. But that would be so much less exciting to implement.

---

263 ExampleWidget Enterprises *RB*

Allowing the player to set a price for a widget on sale, then determining the resulting sales based on consumer demand, and the resulting profit and loss.

Suppose the player is responsible for pricing at Widget Enterprises. Widget production entails a certain fixed cost as well as a cost per unit; and somewhere out in the world there are a number of customers interested in purchasing widgets, but the player starts without knowing what this distribution looks like.

We can express the profits as an equation: the total made by selling

widgets, minus the cost thereof.

The Table of Customers holds the data about customer preferences, and whenever the player selects a widget price, we consult it to determine how many customers in total would be willing to buy at that price.

"Widget Enterprises"

Widget Stand is a room.

A monetary value is a kind of value. $1.99 specifies a monetary value with parts dollars and cents.

Equation - Profit Equation
P = nV - (F + nC)
where P is a monetary value, F is the fixed cost, C is the unit cost, V is a monetary value, and n is a number.

The fixed cost is a monetary value that varies. The fixed cost is $5.00.
The unit cost is a monetary value that varies. The unit cost is $10.66.

Table of Customers

base maximum value
$26.00
$20.00
$15.00
$13.50
$9.00

To decide what number is the units sold at (V - a

```
monetary value):
    let total units be 0;
    repeat through the Table of Customers:
        if V is less than the maximum value entry:
            increase total units by the base entry;
    decide on total units.



Understand "set price to [monetary value]" as setting
price to. Setting price to is an action applying to one
monetary value.



Carry out setting price to:
    let V be the monetary value understood;
    let n be the units sold at the monetary value
understood;
    let P be given by the Profit Equation;
    say "You set the price of your widgets to [V], resulting
in sales of [n] unit[s] and ";
    if P is less than $0.00:
        let L be $0.00 - P;
        say "a loss of [L].";
    otherwise if P is $0.00:
        say "break even.";
    otherwise:
        say "a profit of [P].".



Test me with "set price to $0.00 / set price to $100.00 /
set price to $15.00 / set price to $8.00 / set price to
$25.00 / set price to $14.99".
```

As written this will be a rather dull guessing game for the player; more interesting would be to enhance it into a fuller economic simulator with more control over fixed costs and customer price points.

ExampleFrozen Assets *RB*

A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping.

In our brave new world, everything will have a price, so we had better

spell this out.

```
"Frozen Assets"

Price is a kind of value. $10.99 specifies a price. A thing
has a price. The price of a thing is usually $0.00. After
examining something for sale, say "It can be yours for
[the price of the noun]."
```

Now we assume a simple shopping model in which the player can't take anything without paying for it.

```
Definition: a thing is free if the price of it is $0.00.
Definition: a thing is for sale if it is not free.

Instead of taking something for sale:
    say "You'll have to pay for that."

Before buying something for sale when the money is not
in the wallet:
    say "You're broke." instead.

Before buying something for sale when the money is
free:
    say "You're broke." instead.

Before buying something for sale when the price of the
money is less than the price of the noun:
    say "Your funds do not run to the price of [the noun]."
instead.

Instead of buying something:
```

```
    decrease the price of the money by the price of the
noun;
    say "You fork over [the price of the noun] for [the
noun], leaving yourself with [the price of the money].";
    if the money is free:
        now the money is nowhere;
    now the price of the noun is $0.00;
    now the player is carrying the noun.
```

The player's money object is going to be a bit unusual, because it has value but cannot itself be bought.

```
The player carries a wallet. The wallet contains money.
The price of the money is $4.50. The printed name of
the money is "[price of the money] in cash". Understand
"cash" as the money.


Instead of taking the money:
    say "Best to leave it alone until you need to buy
something."


Instead of buying something free:
    say "[The noun] is yours already."


Instead of buying the money:
    say "The money belongs to you; you buy things with
it."
```

Now we just need something to buy.

```
The Dessert Parlor is a room. "An underlit, over-
crowded room campily furnished with a lot of gilt-frame
mirrors and twinkle lights: it is essentially a brothel of
food. The service is slow at best, and on Saturday
nights glacial. However. The wares on display more
than make up for these trivial inconveniences."
```

```
The vanilla ice cream is an edible thing in the Parlor. The price of the ice cream is $2.45. The description is "In the scale of ice creams, you recognize this as a very inferior vanilla because it has no adjectives in the title."

The raspberry tart is an edible thing in the Parlor. The price of the tart is $4.50. The description is "An almond-laced shell packed with raspberries-under-glaze."

The syllabub is an edible thing in the Parlor. The price of the syllabub is $4.25. The description is "Whipped cream, alcohol, and lime juice, a substance without any redeeming food value whatever."

The espresso cake is an edible thing in the Parlor. The price of the espresso cake is $5.50. The description is "A lethal wedge of purest blackness."

Test me with "inventory / examine syllabub / get syllabub / buy syllabub / drop it / get it / buy raspberry tart".
```

Implementing caloric units for this scenario is left as an exercise for the reader.

Example **Money for Nothing** *RB*

An OFFER price FOR command, allowing the player to bargain with a flexible seller.

```
"Money for Nothing"

Section 1 - Prices and Bargaining
```

Price is a kind of value. $10.99 specifies a price with parts dollars and cents (optional, preamble optional).

A person has a price called wealth. The wealth of the player is $15.

A thing has a price called minimum value. The minimum value of a thing is usually $0.50.

A thing has a price called desired value. The desired value of a thing is usually $5.00.

Offering it for is an action applying to one price and one visible thing.

Understand "offer [price] for [something]" as offering it for.

After taking inventory, say "You have [the wealth of the player]."

Check offering it for:
    if the price understood is greater than the wealth of the player, say "You don't have that kind of cash." instead;
    if the second noun is not carried by someone, say "There's no one in a position to sell you [the second noun]." instead;
    if the second noun is carried by the player, say "[The second noun] is already yours." instead;
    if the minimum value of the second noun is greater than the price understood, say "[The holder of the

```
second noun] cackles disdainfully. 'If yer just here to
insult me you can take your business elsewhere!' he
says." instead;
    if the desired value of the second noun is greater than
the price understood:
        let difference be the desired value of the second
noun minus the price understood;
        let difference be difference divided by two;
        decrease the desired value of the second noun by
difference;
        now the last object offered is the second noun;
        say "'How about [desired value of the second
noun]?' suggests [the holder of the second noun]."
instead;
    otherwise:
        unless the desired value of the second noun is the
price understood:
            say "From the avaricious gleam in the eye of
[the holder of the second noun], you guess you could've
gotten this purchase for less..."



Carry out offering it for:
    increase the wealth of the holder of the second noun
by the price understood;
    decrease the wealth of the player by the price
understood;
    move the second noun to the player.



Report offering it for:
    say "You spend [the price understood], and now you
possess [the second noun]."



When play begins: now right hand status line is "Your
funds: [wealth of the player]".
```

Now, since the man does make counter-offers, it would be reasonable to let the player accept or reject those, as well:

```
The last object offered is a thing that varies.
```

```
Instead of saying yes when the last object offered is
carried by a person (called seller) who is not the player:
    if the seller is not visible:
        continue the action;
    otherwise:
        now the price understood is the desired value of
the last object offered;
        try offering the desired value of the last object
offered for the last object offered.
```

```
Instead of saying no when the last object offered is
carried by a person (called seller) who is not the player:
    if the seller is not visible:
        continue the action;
    otherwise:
        now the last object offered is the player;
        say "You reject the offer firmly."
```

And we borrow just a line or two from a later chapter to take care of some alternate syntax the player might try:

```
Understand "offer [price] to [someone]" as a mistake
("You'll need to specify what you want to buy -- try
OFFER $1000.00 FOR BROOKLYN BRIDGE.").
Understand "offer [someone] [price]" as a mistake
("You'll need to specify what you want to buy -- try
OFFER $1000.00 FOR BROOKLYN BRIDGE.").
```

```
Understand "buy [something]" as a mistake ("You'll have
to name your price: try OFFER $1000.00 FOR
BROOKLYN BRIDGE.").
```

```
Section 2 - The Scenario
```

```
The Flea Market is a room. The crotchety man is a man
in the Market. "A crotchety man here is selling [the list of
things carried by the crotchety man]." The crotchety
man carries a broken television set, a Victorian
rhinestone brooch, and a cracked shaving mug.
```

The minimum value of the brooch is $2.50.

Test me with "offer $0.50 for mug / offer $0.50 to man / offer $6.00 for mug / offer $50.00 for brooch / offer $1.50 for brooch / offer $4.50 for brooch / no / offer $4.50 for brooch / yes".

---

266 **ExampleLemonade** *RB*

Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

Liquids, and all substances that can be mixed or broken off in partial amounts, pose a challenge to model in interactive fiction. The following example is a simple one, but adequate for many scenarios.

We start by assuming that all liquids in the game will always appear in containers. The player can pour liquids from one container to another, and the containers keep track of how full they are and describe themselves appropriately. The player can also refer to containers by content.

Mixture, however, is not allowed, nor is it possible to put liquids on other objects, pour them out on the ground, etc. These ideas would require a more complicated set-up.

"Lemonade"

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The current volume of a fluid container is usually 0.0 fl oz.

Liquid is a kind of value. The liquids are water, milk, lemonade, and iced tea. A fluid container has a liquid.

Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun] at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the noun in rough terms] of [liquid of the noun]."

To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3 and tenths part of amount is less than 7:
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid ounce[s]".

Before printing the name of a fluid container (called the target) while not drinking:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.

After printing the name of a fluid container (called the

```
target) while not examining:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.



Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."



Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.



Understand "drink from [fluid container]" as drinking.



Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within."
instead;
    otherwise:
        decrease the current volume of the noun by 0.2 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "You take a sip of [the liquid of the noun][if the
noun is empty], leaving [the noun] empty[end if]."
```

We have allowed all liquids to be drunk, but it would be possible also to add checking, if we had a game where some liquids were beverages and others were, say, motor oil.

```
Understand the command "fill" as something new.



Understand "pour [fluid container] in/into/on/onto [fluid
container]" as pouring it into. Understand "empty [fluid
container] into [fluid container]" as pouring it into.
```

Understand "fill [fluid container] with/from [fluid container]" as pouring it into (with nouns reversed).

Understand "pour [something] in/into/on/onto [something]" as pouring it into. Understand "empty [something] into [something]" as pouring it into. Understand "fill [something] with/from [something]" as pouring it into (with nouns reversed).

Pouring it into is an action applying to two things.

Check pouring it into:
    if the noun is not a fluid container, say "You can't pour [the noun]." instead;
    if the second noun is not a fluid container, say "You can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second noun:
        if the second noun is empty, now the liquid of the second noun is the liquid of the noun;
        otherwise say "Mixing [the liquid of the noun] with [the liquid of the second noun] would give unsavory results." instead;
    if the noun is empty, say "No more [liquid of the noun] remains in [the noun]." instead;
    if the second noun is full, say "[The second noun] cannot contain any more than it already holds." instead.

Carry out pouring it into:
    let available capacity be the fluid capacity of the second noun minus the current volume of the second noun;
    if the available capacity is greater than the current volume of the noun, now the available capacity is the current volume of the noun;
    increase the current volume of the second noun by available capacity;
    decrease the current volume of the noun by available capacity.

```
Report pouring it into:
    say "[if the noun is empty][The noun] is now empty;
[otherwise][The noun] now contains [current volume of
the noun in rough terms] of [liquid of the noun]; [end if]";
    say "[the second noun] contains [current volume of
the second noun in rough terms] of [liquid of the second
noun][if the second noun is full], and is now full[end if]."
```

This is probably a drier description than we would actually want in our story, but it does allow us to see that the mechanics of the system are working, so we'll stick with this for the example.

Now we need a trick from a later chapter, which allows something to be described in terms of a property it has. This way, the story will understand not only "pitcher" and "glass" but also "pitcher of lemonade" and "glass of milk" -- and, indeed, "glass of lemonade", if we empty the glass and refill it with another substance:

```
Understand the liquid property as describing a fluid
container. Understand "of" as a fluid container.
```

And now the scenario itself:

```
The Porch is a room. The porch swing is an enterable
supporter in the Porch. "An inviting swing hangs here at
the end of the porch, allowing you to enjoy the summer
with a cool beverage, and watch your neighbor Ted
mowing his lawn with the very last manual powerless
lawnmower on the block."

The glass is a fluid container carried by the player. The
liquid of the glass is milk. The current volume of the
glass is 0.8 fl oz.

The pitcher is a fluid container in the Porch. The fluid
capacity of the pitcher is 32.0 fl oz. The current volume
of the pitcher is 20.0 fl oz. The liquid of the pitcher is
lemonade.
```

Ted's Lawn is outside from the Porch. Ted is a man in Ted's Lawn. "Ted has taken off his shirt, but still seems a bit oppressed by the sun." The description of Ted is "He looks hot. In all senses."

After deciding the scope of the player: place Ted in scope.

Instead of doing something to Ted when the player is in the Porch: say "You can't really interact with Ted from this distance, except in the sense of eyeing him surreptitiously."

Instead of giving an empty fluid container to Ted: say "Yes, taunt the poor man, why don't you?"

Instead of giving a fluid container to Ted when the liquid of the noun is milk: say "Ted looks ruefully at the milk. 'Thanks, but I'm lactose-intolerant,' he says."

The block giving rule is not listed in the check giving it to rules.

```
Every turn:
    if Ted is in the location:
        if Ted carries a fluid container (called refreshment):
            try Ted drinking the refreshment;
        otherwise if a random chance of 1 in 3 succeeds:
            say "Ted pushes the ineffective mower over
some dandelions."
```

Instead of someone drinking a fluid container:

```
    if the noun is empty:
        try the person asked giving the noun to the player;
    otherwise:
        decrease the current volume of the noun by 2.0 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "[The person asked] gulps down some [liquid
of the noun]."



After someone giving something to the player:
    say "'Here,' says [the person asked], handing [the
noun] back to you. 'Thanks, I owe you one.'";
    end the story finally.



Test me with "x milk / x lemonade / drink lemonade /
drink milk / pour lemonade into glass / drink milk / x milk
/ drink milk / g / i / fill glass with lemonade / drink
lemonade / drop glass / drink lemonade / pitcher".



Test Ted with "out / give milk to ted / drink milk / g / g / g
/ give glass to ted / in / fill glass with lemonade / out /
give lemonade to ted / wait / z / z / z ".
```

---

267 **ExampleSavannah** *RB*

Using the liquid implementation demonstrated in Lemonade for putting out fires.

Here we build very slightly on the existing liquid implementation to add a puzzle where the player puts out a fire with a bucket of water. Most of the liquid implementation remains the same as before, but now we understand the names of containers according to the liquids they contain.

The new material, pertaining to extinguishing fires, is at the bottom in section 2.

```
"Savannah"
```

Section 1 - Essentials of Liquid

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The current volume of a fluid container is usually 0.0 fl oz.

Liquid is a kind of value. A fluid container has a liquid.

```
Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun] at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the noun in rough terms] of [liquid of the noun]."

To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3 and tenths part of amount is less than 7:
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase
```

```
amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid
ounce[s]".


Before printing the name of a fluid container (called the
target) while not drinking:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.


After printing the name of a fluid container (called the
target) while not examining:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.


Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."


Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.


Understand "drink from [fluid container]" as drinking.


Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within."
instead;
    otherwise:
        decrease the current volume of the noun by 0.2 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
```

```
        say "You take a sip of [the liquid of the noun][if the
noun is empty], leaving [the noun] empty[end if]."



Understand the command "fill" as something new.



Understand "pour [fluid container] in/into/on/onto [fluid
container]" as pouring it into. Understand "empty [fluid
container] into [fluid container]" as pouring it into.
Understand "fill [fluid container] with/from [fluid
container]" as pouring it into (with nouns reversed).



Understand "pour [something] in/into/on/onto
[something]" as pouring it into. Understand "empty
[something] into [something]" as pouring it into.
Understand "fill [something] with/from [something]" as
pouring it into (with nouns reversed).



Pouring it into is an action applying to two things.



Check pouring it into:
    if the noun is not a fluid container, say "You can't pour
[the noun]." instead;
    if the second noun is not a fluid container, say "You
can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly
pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second
noun:
        if the second noun is empty, now the liquid of the
second noun is the liquid of the noun;
        otherwise say "Mixing [the liquid of the noun] with
[the liquid of the second noun] would give unsavory
results." instead;
    if the noun is empty, say "No more [liquid of the noun]
remains in [the noun]." instead;
    if the second noun is full, say "[The second noun]
cannot contain any more than it already holds." instead.
```

```
Carry out pouring it into:
    let available capacity be the fluid capacity of the
second noun minus the current volume of the second
noun;
    if the available capacity is greater than the current
volume of the noun, now the available capacity is the
current volume of the noun;
    increase the current volume of the second noun by
available capacity;
    decrease the current volume of the noun by available
capacity.


Report pouring it into:
    say "[if the noun is empty][The noun] is now empty;
[otherwise][The noun] now contains [current volume of
the noun in rough terms] of [liquid of the noun]; [end if]";
    say "[the second noun] contains [current volume of
the second noun in rough terms] of [liquid of the second
noun][if the second noun is full], and is now full[end if]."


Understand the liquid property as describing a fluid
container. Understand "of" as a fluid container.


Section 2 - Putting Out Fires


The Beach is a room. "The Atlantic stretches east to the
horizon, though it is at low tide at the moment. It is
dawn: time to pack up and go home."
```

We will skip implementing the Pacific ocean itself, though the example Lakeside Living shows how to incorporate large bodies of water into our liquid simulation.

```
The liquids are seawater. [We could include others, but
for the moment...]
```

```
Instead of drinking a fluid container when the liquid of
the noun is seawater:
    say "Blech!"


The bucket is a fluid container carried by the player. The
liquid of the bucket is seawater. The current volume of
the bucket is 64.0 fl oz.


The fire is a fixed in place thing in the beach. "A low fire
crackles here, left over from an attempt at s'mores much
earlier in the evening."


Instead of touching or rubbing or taking the fire, say
"You're not such a glutton for punishment."


Instead of pouring something into the fire:
    now the fire is nowhere;
    now the current volume of the noun is 0.0 fl oz;
    say "[The second noun] goes out in a great hiss."


Test me with "drink seawater / pour seawater on fire / x
bucket / i".
```

This is still a specific implementation: if we wanted to weave liquids together with a full-scale burning model (as in "In Fire or in Flood"), where pretty much any object in the game can be flaming (currently on fire) or damp (extinguished), we might generalize our rule to

```
Instead of pouring something into a flaming thing:
    now the second noun is damp;
    now the current volume of the noun is 0.0 fl oz;
    say "[The second noun] goes out in a great hiss."
```

Of course, the merging of fire and liquids also raises the possibility of gasoline and explosives, of heating and boiling liquids, etc.: as always, it's wise to incorporate a simulation that is only as detailed as the game's

interactions really justify.

---

268 ExampleDepth *RB*

Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

In the following, we pretend that every item has a cuboidal shape. Every thing has a length, width and depth, while a "measured container" also has interior dimensions. (Thus a 10x10x10 container with 1cm-thick sides might have interior dimensions 9x9x9.)

```
"Depth"

A length is a kind of value. 10 cm specifies a length. An
area is a kind of value. 10 sq cm specifies an area. A
length times a length specifies an area. A volume is a
kind of value. 10 cu cm specifies a volume. A length
times an area specifies a volume.

A thing has a length called height. A thing has a length
called width. A thing has a length called depth. The
height of a thing is usually 10 cm. The width of a thing is
usually 10 cm. The depth of a thing is usually 10 cm.

To decide what volume is the exterior volume of (item -
a thing):
    let base area be the height of the item multiplied by
the width of the item;
    let base volume be the base area multiplied by the
depth of the item;
    decide on the base volume.
```

In order to see how these shapes might fit together spatially, we need to work out the three dimensions in order of size. (If we were only dealing with portable objects, we could simply insist that the length always be greater than the width which in turn must be greater than the depth,

because we could always turn them over in our hands until this was so: but some of the things we deal with may be fixed in place.) A clever way to do this might be to put them in a table of three rows and sort it, but we will write the calculation out longhand:

```
To decide what length is the largest dimension of (item -
a thing):
    let long side be the height of item;
    if the width of the item is greater than the long side,
now the long side is the width of the item;
    if the depth of the item is greater than the long side,
now the long side is the depth of the item;
    decide on the long side.
```

```
To decide what length is the middling dimension of (item
- a thing):
    let longer side be the height of item;
    let shorter side be the width of item;
    if the width of the item is greater than the height of the
item:
        let shorter side be the height of item;
        let longer side be the width of item;
    if the depth of the item is greater than the longer side,
decide on the longer side;
    if the depth of the item is less than the shorter side,
decide on the shorter side;
    decide on the depth of the item.
```

```
To decide what length is the shortest dimension of (item
- a thing):
    let short side be the height of item;
    if the width of the item is less than the short side, now
the short side is the width of the item;
    if the depth of the item is less than the short side, now
the short side is the depth of the item;
    decide on the short side.
```

When testing this example, the author made use of the following: it's no longer needed, but may be useful to anyone else planning elaborations.

```
To test the dimensions of (item - a thing):
```

```
    say "[the item] - height [height of the item], width
[width of the item], depth [depth of the item].";
    say "largest side [largest dimension of the item],
middling [middling dimension of the item], smallest
[shortest dimension of the item]."
```

We now introduce a new kind: a measured container, which not only has exterior dimensions - the height, width and depth which every thing now has - but also interior measurements. A convenient way to do calculations with the hollow interior is to regard it as if it were a solid shape in its own right, and we do this with the aid of something out of world, which the player never sees: the "imaginary cuboid", which is made into the shape of whatever measured container's interior is being thought about.

```
A measured container is a kind of container. A
measured container has a length called interior height.
A measured container has a length called interior width.
A measured container has a length called interior depth.


There is an imaginary cuboid.


To imagine the interior of (receptacle - a measured
container) as a cuboid:
    now the height of the imaginary cuboid is the interior
height of the receptacle;
    now the width of the imaginary cuboid is the interior
width of the receptacle;
    now the depth of the imaginary cuboid is the interior
depth of the receptacle.


To decide what volume is the interior volume of
(receptacle - a measured container):
    imagine the interior of the receptacle as a cuboid;
    decide on the exterior volume of the imaginary
cuboid.
```

If we assume that we could always pack items into a measured container with perfect ease, never wasting any space, then the only volume constraint will be that the total volume of the contents must not exceed the volume of the inside of the container. So we need to calculate the

available volume.

```
To decide what volume is the available volume of
(receptacle - a measured container):
    let the remaining space be the interior volume of the
receptacle;
    repeat with item running through things in the
receptacle:
        decrease the remaining space by the exterior
volume of the item;
    if the remaining space is less than 0 cu cm, decide on
0 cu cm;
    decide on the remaining space.
```

If we only constrained volume, a 140 cm-long fishing rod could fit into a 12 cm by 12 cm compact disc box. So we also insist the basic shape must fit, in some orientation perpendicular to one of the sides (i.e.: we can turn the item over in any of its three sides, but not turn it diagonally or wedge it in at a tilt). This requires the longest side of the item to be less than the longest side of the receptacle, and the middle-length side, and also the shortest side. The number of these conditions to fail gives us a clue as to how we can best describe the reason why the shape won't squeeze in.

```
Check inserting something (called the item) into a
measured container (called the receptacle):
    if the exterior volume of the item is greater than the
interior volume of the receptacle, say "[The item] will
never fit inside [the receptacle]." instead;
    if the exterior volume of the item is greater than the
available volume of the receptacle, say "[The item] will
not fit into [the receptacle] with [the list of things in the
receptacle]." instead;
    imagine the interior of the receptacle as a cuboid;
    if the largest dimension of the item is greater than the
largest dimension of the imaginary cuboid, say "[The
item] is too long to fit into [the receptacle]." instead;
    if the middling dimension of the item is greater than
the middling dimension of the imaginary cuboid, say
"[The item] is too wide to fit into [the receptacle]."
instead;
    if the shortest dimension of the item is greater than
the shortest dimension of the imaginary cuboid, say
"[The item] is too bulky to fit into [the receptacle]."
instead.
```

And finally a situation to try out these rules.

```
The Cubist Lab is a room. "A laboratory which, as the
art critic Louis Vauxcelles said about Braque's paintings
in 1908, is full of little cubes: everyday objects rendered
as if cuboidal."


The box is a measured container. The interior height is
10 cm. The interior depth is 5 cm. The interior width is 6
cm. The player carries the box.


A pebble is a kind of thing. The height is usually 2 cm.
The depth is usually 2 cm. The width is usually 2 cm.
The player carries 25 pebbles.


A red rubber ball is carried by the player. The depth is 5
cm. The width is 5 cm. The height is 5 cm.
An arrow is carried by the player. The height is 40 cm.
The width is 1 cm. The depth is 1 cm.
A crusty baguette is carried by the player. The height is
80 cm. The width is 4 cm. The depth is 5 cm.
A child's book is carried by the player. The height is 1
cm. The width is 9 cm. The depth is 9 cm.
A featureless white cube is carried by the player. The
height is 6 cm. The width is 6 cm. The depth is 6 cm.


Test me with "put arrow in box / put book in box / put
cube in box / put ball in box / put baguette in box / put
pebbles in box".
```

Several warnings about this. First, the numbers can't go very high (if the Settings for the project set the story file format to the Z-machine): while the volume can in theory go to 32,767, in practice this equates to an object 32 cm on a side, which is not very large. One way to avoid this is to use the Glulx format, allowing for sizes in excess of 10 m on a side: or we could simply scale the dimensions to suit our purposes, using a decimeter (10 cm) as the basic unit of measurement, for instance.

Second, the system will require a height, width, and depth for every

portable object in the game, which is a large commitment to data entry; it may become tiresome. So it is probably not worth bothering with this kind of simulation unless it is going to be genuinely significant.

---

269 **ExampleFabrication** *RB*

A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

When we make a new kind of value, the new named values can themselves have properties. That is convenient because, for instance, we might want to associate a material (itself the property of an object) with certain features, such as price.

```
"Fabrication"

Section 1 - Procedure

A material is a kind of value. The materials are silk,
velvet, cotton, and wool.

Price is a kind of value. $1.99 specifies a price.

Area is a kind of value. 5 sq yards specifies an area.

Cost is a kind of value.. $1.99 per sq yard specifies a
cost. A cost times an area specifies a price.

A material has a cost.
```

The cost of silk is usually $5.75 per sq yard. The cost of velvet is usually $9.50 per sq yard. The cost of cotton is usually $2.29 per sq yard. The cost of wool is usually $4.75 per sq yard.

A pattern is a kind of thing. A pattern has a material. A pattern has an area. A pattern has a price. The price of a pattern is usually $9.99. Understand "pattern" as a pattern. Understand "patterns" as the plural of a pattern.

After printing the name of a pattern:
    if planning:
        do nothing;
    otherwise:
        say " pattern".

To decide what price is the material price of (chosen item - pattern):
    let C be the cost of the material of the chosen item;
    let A be the area of the chosen item;
    decide on C * A.

To decide what price is the overall price of (chosen item - pattern):
    let P be the price of the chosen item;
    let M be the material price of the chosen item;
    decide on P + M.

Understand "plan [material] [pattern]" as planning it for.

Planning it for is an action applying to one material and one thing.

Carry out planning it for:
    now the material of the second noun is the material understood.

Report planning it for:
    say "You lay plans for a [material understood] [second noun], running [material price of the second noun] for materials and [price of the second noun] for the pattern itself, for a total of [overall price of the second noun]."

Section 2 - Scenario

Joanne's Fabrics is a room. Joanne's Fabrics contains a pattern bin.

The cape is a pattern. The material of the cape is velvet. The area of the cape is 9 sq yards.

The bodice is a pattern. The material of the bodice is silk. The area of the bodice is 2 sq yards. The price of the bodice is $11.99.

The cape and the bodice are in the pattern bin.

Test me with "plan silk bodice / plan velvet bodice / plan velvet cape / plan wool cape".

The Speed of Thought
Describing scientifically-measured objects in units more familiar to the casual audience.

Suppose that we have a number of objects in the game that are sized in some conventional unit (such as meters), but which we would like to describe in slightly less formal terms. To do this, we will start with measurements as defined in the built-in extension Metric Units, so we don't have to recreate all these.

We'll add our own set of "conceptual units" -- things we're familiar with in real life. As we'll see below, Inform will automatically choose a unit of the right order to express a given distance if we tell it to print a length "in conceptual units".

Note: the following will compile only if you have settings set for Glulx. (To change this, go to the Settings panel and click on the Glulx option.) The Glulx virtual machine is capable of handling larger numbers than the Z-machine.

```
"The Speed of Thought"


Section 1 - Procedure


Include Metric Units by Graham Nelson.


1 quarter (in conceptual units, in quarters, singular) or 2
quarters (in conceptual units, in quarters, plural)
specifies a length equivalent to 24mm.
1 pencil (in conceptual units, in pencils, singular) or 2
pencils (in conceptual units, in pencils, plural) specifies
a length equivalent to 18cm.
1 bathtub (in conceptual units, in bathtubs, singular) or 2
bathtubs (in conceptual units, in bathtubs, plural)
specifies a length equivalent to 152cm.
1 Olympic swimming pool (in conceptual units, in
Olympic swimming pools, singular) or 2 Olympic
swimming pools (in conceptual units, in Olympic
swimming pools, plural) specifies a length equivalent to
50 meters.
```

1 Empire state building (in conceptual units, in Empire State buildings, singular) or 2 Empire State buildings (in conceptual units, in Empire State buildings, plural) specifies a length equivalent to 443m.

1 credit card (in conceptual units, in credit cards, singular) or 2 credit cards (in conceptual units, in credit cards, plural) specifies an area equivalent to 46 sq cm.
1 letter sheet (in conceptual units, in letter sheets, singular) or 2 letter sheets (in conceptual units, in letter sheets, plural) specifies an area equivalent to 603 sq cm.
1 queen-sized mattress (in conceptual units, in queen-sized mattresses, singular) or 2 queen-sized mattresses (in conceptual units, in queen-sized mattresses, plural) specifies an area equivalent to 3 square meters.
1 football field (in conceptual units, in football fields, singular) or 2 football fields (in conceptual units, in football fields, plural) specifies an area equivalent to 5351 square meters.

Understand "report [something]" as reporting. Reporting is an action applying to one thing.

Check reporting:
    if the noun is not a fact:
        say "The public doesn't want to hear about [the noun]." instead.

Carry out reporting:
    now the noun is nowhere.

Report reporting:
    if the extent of the noun is greater than 0mm and the surface of the noun is greater than 0 sq cm:
        contextualize "'[The noun] has a length of [about] [extent of the noun in conceptual units] and an area of [about] [surface of the noun in conceptual units].'";
    otherwise if the extent of the noun is greater than

```
0mm:
        contextualize "'[The noun] has a length of [about]
[extent of the noun in conceptual units].'";
    otherwise if the surface of the noun is greater than 0
sq cm:
        contextualize "'[The noun] has an area of [about]
[surface of the noun in conceptual units].'";
    otherwise:
        say "'[The noun] is... pretty hard to imagine,' you
say weakly. That's not going to go over well."



To say about:
    say "[one
of]roughly[or]about[or]around[or]approximately[at
random]";



To contextualize (chosen information - text):
    say "[one of]You turn to the camera and speak:[or]
[or]Turning to another camera angle, you add:[or]
[stopping] ";
    say "[chosen information] ";
    say "[one of][line break][or]Right now the station will
be cutting over to a visual of that.[or][line break][or]Pity
the kids in audiovisual who have to scare that image
together in a hurry.[or]You smile brightly.[stopping]";



Section 2 - Scenario



The Science Journalism Desk is a room. "From here
you, the Science Anchor, have the privilege of reporting
the latest and most fascinating stories to an eager
public."



After looking:
    try thinking.
```

Instead of thinking:
    say "Currently you have to report on the International Space Station. Your story could include [the list of facts carried by the player]."

Instead of taking inventory:
    say "It looks foolish to be fiddling with your possessions on camera."

Instead of dropping a fact:
    say "You decide to omit [the noun] from your lineup.";
    now the noun is nowhere.

A fact is a kind of thing. Every fact is carried by the player. A fact has a length called the extent. A fact has an area called the surface.

The experiment module is a fact. The extent is 1116cm.
The logistics module is a fact. The extent is 421cm.
The solar array is a fact. The surface is 375 sq m. The extent is 58m.
An individual solar cell is a fact. The surface is 8 sq cm.
The orbit height is a fact.

Report reporting the orbit height:
    contextualize "'The station orbits at heights between [about] [278km in conceptual units] and [460km in conceptual units] above the earth.'" instead.

Every turn:
    if the player carries no facts:
        say "And that's all! The channel cuts to weather.";
        end the story saying "Time for lunch".

Test me with "report experiment module / report logistics / report height / report array / report solar cell".

Chapter 16: Tables

6.9. Blank rows; *§16.10. Adding and removing rows*; *§16.11. Sorting*; *§16.12. Listed in...*; *§16.13. Topic*

Contents of *Writing with Inform*

Chapter 15: Numbers and Equations

Chapter 17: Understanding

Indexes of the examples

§16.1. Laying out tables

When printed books need to display detailed information in a systematic way, they break off from running text and print a table instead. Inform does the same. Here is a typical example:

```
Table 2.1 - Selected Elements

Atomic number	Symbol	Atomic weight	Element
1	H	1.008	"Hydrogen"
26	Fe	55.845	"Iron"
55	Cs	132.905	"Caesium"
92	U	238.029	"Uranium"
```

After the two titling lines, each line represents one row in the table, and entries on a line must be separated by at least one tab character. A table must occupy a single whole paragraph, with no skipped lines or missing entries.

The top line is a title, the first word of which must be the word 'Table'. We can then either give a table number (this need not actually be a number: Table C2, or some such, would be fine), or give a name, or both - as in this case. The possible titling formats are:

```
Table 2.3
Table of Population Statistics
Table 2.3 - Population Statistics
```

In the last example we could call the table either "Table 2.3" or "Table of Population Statistics".

Each column then has a name, and the contents must all be the same kind of value. In the elements table the "Symbol" column contains only text, for instance, and the "Atomic weight" column contains only numbers. Any kinds of value will do, so long as all the entries in the column are mutually compatible. (For instance, mixing rooms and things in a single column would be fine, as these can be reconciled, but mixing numbers and rooms would not.)

Start of Chapter 16: Tables

Back to Chapter 15: Numbers and Equations: §15.20. Multiplication of units

Onward to §16.2. Looking up entries

---

§16.2. Looking up entries

The simplest way to access the information inside tables is to ask explicitly for it, specifying the row number, the column name and what table is to be consulted. So, given our example table

Table 2.1 - Selected Elements

Symbol	Atomic weight
"H"	"Hydrogen"
"Zn"	"Zinc"
"U"	"Uranium"

we can write the following description:

```
symbol in row 3 of the Table of Selected Elements
```

to produce the value "Zn". Or the following will run off some chemical data:

```
repeat with N running from 1 to the number of rows in the Table of
Selected Elements:
    say "The atomic weight of [element in row N of the Table of
Selected Elements] is [atomic weight in row N of the Table of
Selected Elements]."
```

The result of which will be:

```
The atomic weight of Hydrogen is 1.
The atomic weight of Iron is 56.
The atomic weight of Zinc is 65.
The atomic weight of Uranium is 238.
```

Note that the first row in a table is row number 1, and that the last can be found with the phrase:

number of rows in/from (table name) **... number**

This phrase produces the number of rows (including any blank rows) in the given table. Example:

```
number of rows in the Table of Selected Elements
```

Start of Chapter 16: Tables
Back to §16.1. Laying out tables
Onward to §16.3. Corresponding entries

§16.3. Corresponding entries

Continuing our example of the elements:

Table 2.1 - Selected Elements

Symbol	Atomic weight
"H"	1
"Hydrogen"	

26 "Fe"
30 "Zinc"
92 "Uranium"

If we want to know the atomic number of Uranium, say, it seems artificial to have to talk about the particular row number where the information happens to be. So we are also allowed to cross-reference, like so:

```
the atomic number corresponding to a symbol of "Fe" in the Table of
Selected Elements
```

This results in 26, and similarly

```
the symbol corresponding to an atomic number of 26 in the Table of
Selected Elements
```

results in "Fe". But we have to be careful:

```
the element corresponding to an atomic number of 27 in the Table of
Selected Elements
```

This is not allowed (it produces an error at run-time), because there is no row with atomic number 27 in this rather limited table. We can check this in advance with the condition:

```
if there is an element corresponding to an atomic number of 27 in the
Table of Standard Elements ...
```

Or more simply:

```
if there is an atomic number of 27 in the Table of Standard Elements
...
```

The condition "if there is..." can be used with any reference to a table entry: for instance, "if there is a symbol in row 5 of the Table of Standard Elements" would be false, because there are only four rows.

Example 271: ★★★ **Dubai**
An elevator which connects any of 27 floors in a luxury hotel.

---

§16.4. Changing entries

Here is another rather definitive, immutable-looking table:

Table 4 - Recent Monarchs

Name	Accession
"Anne"	1702
"George I"	1714
"George II"	1727
"George III"	1760
"George IV"	1820
"William IV"	1830
"Victoria"	1837
"Edward VII"	1901
"George V"	1910
"Edward VIII"	1936
"George VI"	1936
"Elizabeth II"	1952

But table entries can be changed as freely as variables: that is, any value can be entered so long as it has the right kind. We cannot put a dynasty into the "Name" column, or text in the "Accession" column. The phrase needed is "now ... is ...", just as it is for properties or variables:

Dynasty is a kind of value. The dynasties are Stuart, Hanover, Saxe-Coburg-Gotha and Windsor.

The Table Office is a room. The Succession is in the Table Office.

```
"The Succession, a ponderous list of English monarchs, takes pride
of place."


Instead of examining the Succession:
    say "The Succession List runs as follows...";
    repeat with N running from 1 to the number of rows in the Table of
Recent Monarchs:
        say "[accession in row N of Table 4]: [name in row N of Table 4]
([family in row N of Table 4])."


Instead of attacking the Succession:
    now the family corresponding to an accession of 1720 in the Table
of Recent Monarchs is Stuart;
    now the name in row 4 of the Table of Recent Monarchs is
"Graham I";
    now the name in row 5 of the Table of Recent Monarchs is
"Trixibelle IV";
    say "You deface the English succession, making suitable
amendments with a quill pen. Considering it is supposed to be
mightier than the sword the effect is a little disappointing."


Test me with "examine succession / attack it / examine it".
```

Once we start changing tables, it sometimes becomes useful to check what they contain.

showme the contents of (table name)

This phrase prints a crude but sometimes useful display on screen of the current contents of the named table. It's intended for authors to see when testing, not for players of the finished version to see.

say "[current table row]"

This text substitution produces a crude but sometimes useful listing of the entries in the currently chosen table row.

say "[row (number) **in**/from **table** (table name)**]"**

This text substitution produces a crude but sometimes useful listing of the entries in the specified row.

say "[(column name) **in**/from **table** (table name)**]"**

This text substitution produces a crude but sometimes useful listing of the entries in the specified column.

Start of Chapter 16: Tables

Back to §16.3. Corresponding entries

Onward to §16.5. Choosing rows

§16.5. Choosing rows

The following would be one way to print out a list of recent Kings and Queens:

```
To list the succession:
say "The Succession List runs as follows...";
repeat with N running from 1 to the number of rows in the Table of
Recent Monarchs:
    say "[accession in row N of the Table of Recent Monarchs]: [name
in row N of the Table of Recent Monarchs] ([family in row N of the
Table of Recent Monarchs])."
```

This works, but is repetitive. We often want to work on a single row for a while, either to change things or think about the contents, and it is tiresome to keep specifying the row over and over again. The following shorthand provides some relief:

choose a/the/-- **row** (number) **in**/from (table name)

This phrase selects the row with the given number. Row numbers in a table start from 1, so

```
choose row 1 from the Table of Recent Monarchs
```

selects the top row.

That allows us to improve the loop:

```
To list the succession:
    say "The Succession List runs as follows...";
    repeat with N running from 1 to the number of rows in the Table of
Recent Monarchs:
        choose row N in the Table of Recent Monarchs;
        say "[accession entry]: [name entry] ([family entry]).";
```

Actually, as we'll see in the next section, this kind of loop is needed so often that there's a shorthand wording for it.

Note that since "accession" is a column name, "accession entry" means the entry in that column of the currently chosen row. This notation can only be used if a "choose" has certainly already happened, and it is a good idea to make that choice somewhere close by in the source code (and certainly in the same rule or phrase definition) for the sake of avoiding errors. We can also choose rows by specifying something about them, like so:

choose a/the/-- **row with** (table column) **of** (value) **in**/from (table name)

This phrase selects the first row, working down from the top of the given table, in which the given column has the given value. Example:

```
choose row with a name of "Victoria" in the Table of Recent
Monarchs;
```

A run-time problem message is produced if the value isn't found anywhere in that column.

Sometimes it will happen that a column's name clashes with the name of something else: for instance, if we call a column "apples" but we also have a kind called "apple", so that the word "apples" could mean either some fruit or the column. Inform will generally prefer the former meaning as more likely. In case of such trouble, we can simply refer to "the apples column" rather than just "the apples": for instance, "choose row with an apples column of..." rather than "choose row with an apples of..."

We can also choose a row quite at random:

choose a/the/-- **random row in**/from (table name)

This phrase makes a uniformly random choice of non-blank rows in the given table. Note that although a table always has at least one row, it can't be guaranteed that it always has a non-blank row, so it's possible for this to fail: if it does, a real-time problem message is thrown.

Start of Chapter 16: Tables
Back to §16.4. Changing entries
Onward to §16.6. Repeating through tables

§16.6. Repeating through tables

We very often want to run through a table doing something to, or with, each row in turn, so a special loop is provided for this. Rather than having to write all this out:

```
To list the succession:
    say "The Succession List runs as follows...";
    repeat with N running from 1 to the number of rows in the Table of
Recent Monarchs:
        choose row N in the Table of Recent Monarchs;
        say "[accession entry]: [name entry] ([family entry])."
```

We can simply use this instead:

repeat through (table name)**:**

This phrase causes the block of phrases following it to be repeated once for each row in the given table, choosing each row in turn, from top to bottom. Blank rows are skipped. Example:

```
To list the succession:
    say "The Succession List runs as follows...";
    repeat through the Table of Recent Monarchs:
        say "[accession entry]: [name entry] ([family entry])."
```

Note that there is no loop variable here, unlike in other forms of "repeat", because it's the choice of row which keeps track of how far we have got.

We can alternatively go backwards:

repeat through (table name) **in reverse order:**

This phrase causes the block of phrases following it to be repeated once for each row in the given table, choosing each row in turn, from bottom to top. Blank rows are skipped.

More often we want a sequence which is neither forwards nor backwards, but which depends on the actual values in the table.

repeat through (table name) **in** (table column) **order:**

This phrase causes the block of phrases following it to be repeated once for each row in the given table, choosing each row in turn, in order of the values in the given column. Blank rows are skipped. Example:

```
repeat through the Table of Recent Monarchs in name
order: ...
repeat through the Table of Recent Monarchs in accession
order: ...
```

work through the same table in rather different orders. The sequence is lower to higher (small numbers to high numbers, A to Z, and so on); insert "reverse" after "in" to reverse this.

repeat through (table name) **in reverse** (table column) **order:**

This phrase causes the block of phrases following it to be repeated once for each row in the given table, choosing each row in turn, in order of the values in the given column. Blank rows are skipped. Example:

```
repeat through the Table of Recent Monarchs in reverse
name order: ...
repeat through the Table of Recent Monarchs in reverse
accession order: ...
```

work through the same table in rather different orders. The sequence is higher to lower (high numbers to small numbers, Z to A, and so on); delete the "reverse" after "in" to reverse this.

In a loop like this, the data is not searched very efficiently, which is fine for modest-sized tables like the examples in this chapter, but might be a problem for much larger tables: see the later section on sorting.

These definitions mentioned blankness several times, and that's the topic to cover in the next section.

See **Sorting** *for reordering a table to put it into increasing or decreasing order of the entries in any column*

Start of Chapter 16: Tables

Back to §16.5. Choosing rows

Onward to §16.7. Blank entries

Example 272: **Port Royal 4** A cell window through which the player can see people who were in Port Royal in the current year of game-time.

---

§16.7. Blank entries

We are allowed to leave certain entries blank (perhaps to be filled in later, perhaps not) by writing "--" instead of the relevant value:

Table 2.1 - Selected Elements

Atomic number	Element	Symbol	Atomic weight
1	Hydrogen	H	1
26	Iron	Fe	55.8
30	Zinc	Zn	65.4
92	Uranium	U	238

In effect, blank entries don't exist. "--" is not a value, but only a hole where a value might be. It can be useful to check for this:

if there is (a table entry)**:**

This condition is true if the entry referred to exists, that is, that is, the space for it in the table is not blank. Examples:

```
if there is a symbol corresponding to an atomic number of
30 in the Table of Standard Elements ...
if there is an atomic number in row 2 of the Table of
Standard Elements ...
```

if there is no (a table entry)**:**

This condition is true if the entry referred to does not exist, that is, the space for it in the table is blank. Examples:

```
if there is no symbol corresponding to an atomic number of
30 in the Table of Standard Elements ...
if there is no atomic number in row 2 of the Table of
Standard Elements ...
```

Start of Chapter 16: Tables
Back to §16.6. Repeating through tables
Onward to §16.8. Blank columns

§16.8. Blank columns

An entire column of blank entries "--" is problematic:

Table 2 - Selected Elements

Atomic number	Symbol	Name	Atomic weight
1	H	Hydrogen	1.00794
26	Fe	Iron	55.845

30 "Zinc"
92 "Uranium"

Inform is unable to work out what kind of value should go into the "atomic weight" column here, since it has no examples to guess from. We can get around this by writing in the name of a kind of value:

Table 2 - Selected Elements

Atomic number | Atomic weight
1 "Hydrogen"
26 "Iron"
30 "Zinc"
92 "Uranium"

That top entry in the "atomic weight" column is also blank, but now Inform knows that anything put into the column in future will be a number.

If there are many rows, and perhaps several blank columns, it would become very tedious to have to keep typing out "--". So this is optional *at the end of a row*: it remains compulsory for a blank value appearing in between two values which aren't blank. This is the general idea:

Table 2 - Selected Elements

Atomic number | Atomic weight | Symbol | Density | Boiling point | Melting point | Specific heat capacity | Thermal conductivity | Abundance | Discovered by | Discovery year
1 "Hydrogen"
26 "Iron"
30 "Zinc"
92 "Uranium"

Start of Chapter 16: Tables
Back to §16.7. Blank entries
Onward to §16.9. Blank rows

§16.9. Blank rows

There is no difficulty about entirely blank rows: or rather, the only difficulty is once

again that they are boring to type out. We can avoid the necessity by appending "with ... blank rows" at the foot of the table:

```
Table 2 - Selected Elements

Atomic weight
Element (text)
1 "Hydrogen"
26 "Iron"
32 "Zinc"
92 "Uranium"
with 3 blank rows
```

(These words cannot be placed in between rows, but only at the bottom.) And indeed the table can start out completely empty:

```
Table 3 - Undiscovered Periodic Table

Atomic weight (a number)
Element (text)
with 92 blank rows
```

Blank rows are useful because they enable us to add new data to a table. In effect, they are invisible when not used. A repeat loop like

```
repeat through Table 3:
    ...
```

automatically skips blank rows, so it would initially do nothing at all. Similarly, choosing a "random" row will never choose a blank one.

A convenient way to test if a table contains non-blank rows is to use the built-in adjectives "empty" and "non-empty". So:

```
if the Undiscovered Periodic Table is empty, ...
```

tests whether all of its rows are blank; if even one cell contains a value then the table is "non-empty".

Start of Chapter 16: Tables

Back to §16.8. Blank columns

Onward to §16.10. Adding and removing rows

Example 273: ★ **If It Hadn't Been For...** A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

§16.10. Adding and removing rows

Writing in new rows is simple, once we can find space for them:

choose a/the/-- **blank row in**/from (table name)

This phrase chooses a row in the given table which is currently blank under every column. A run-time problem message is issued if no rows are blank. Example:

```
choose a blank row in Table 3;
now element entry is "Fluorine";
now symbol entry is "F";
now atomic number entry is 9;
now atomic weight entry is 19;
```

To avoid problem messages, it can be important to worry about free space. To that end we can not only find the number of rows (as we have already seen) but also the number currently blank and not blank:

number of blank rows in/from (table name) **... number**

This phrase produces the number of rows in the given table which are entirely blank (that is, blank under every column).

number of filled rows in/from (table name) **... number**

This phrase produces the number of rows in the given table which are not entirely blank (that is, at least one column has a value in this row).

"Filled" here really means "non-blank": a row can be filled in this sense even if only one of its values exists. Since every row is either blank or filled, it must be true that:

```
the number of blank rows in Table 3
the number of filled rows in Table 3
```

add up to "the number of rows in Table 3".

We've seen that blank entries can be filled with values using "now":

```
now symbol entry is "F";
```

But the same method can't be used to put blanks back, since a blank is not a value. Instead:

blank out (a table entry)

This phrase replaces the entry referred to with a blank, erasing any value previously stored there. Example:

```
choose row 1 in the Table of Fish Habitats;
blank out the salinity entry;
```

These more destructive phrases need a steady hand:

blank out the whole row

This phrase replaces the currently chosen row with blanks, erasing any value previously stored under any of the columns. Example:

```
choose row 1 in the Table of Fish Habitats;
```

```
blank out the whole row;
```

blank out the whole (table column) **column in** (table)

This phrase replaces the currently chosen column with blanks, erasing any value previously stored in any of the rows. Example:

```
blank out the whole salinity column in the Table of Fish
Habitats;
```

blank out the whole of (table)

This phrase replaces every row of the currently chosen table with blanks, erasing any value previously stored anywhere in it. Example:

```
blank out the whole of the Table of Fish Habitats;
```

This is only really useful when a Table is being used to hold working space for some calculation or other.

Start of Chapter 16: Tables

Back to §16.9. Blank rows

Onward to §16.11. Sorting

Example 274: ★★ **Odyssey** A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

§16.11. Sorting

The three ways to sort a table correspond loosely to the three different orders in

which tables can be repeated through. First:

sort (table name) **in random order**

This phrase rearranges the rows of the given table so that the non-blank rows occur at the top, in a uniformly random order, and any blank rows at the bottom. Example:

```
sort the Table of Recent Monarchs in random order;
```

Secondly:

sort (table name) **in** (table column) **order**

This phrase rearranges the rows of the given table so that the non-blank rows occur at the top, so that the given column has ascending order, and any blank rows at the bottom. Example:

```
sort the Table of Recent Monarchs in accession order;
```

Ascending order means 1 up to 10, say, or A up to Z, with blank values coming last.

sort (table name) **in reverse** (table column) **order**

This phrase rearranges the rows of the given table so that the non-blank rows occur at the top, so that the given column has descending order, and any blank rows at the bottom. Example:

```
sort the Table of Recent Monarchs in reverse name order;
```

Descending order means 10 down to 1, say, or Z down to A, with blank values coming last.

How sorting is done depends on the contents of the column being sorted on. If it

holds numbers then numerical order is used, with 2 coming before 7, and so on. (And similarly for real numbers, though the existence of infinities makes this more interesting.) If times are sorted then they are sorted from midnight to midnight, following the "is greater than" relation, not with 4 AM as the zero point, as with "is after".

If text is sorted then alphabetical order is used, though this doesn't always come out the way you might expect, because upper case and lower case letters are treated as different: A-Z come before a-z, and accented letters such as é come after the regular alphabet. (What's happening here is that Inform is sorting on raw character values, not performing the full Unicode collation algorithm, which would be too slow at run-time.)

Note that blank values will always be placed below non-blank ones, and entirely blank rows last of all. This is true even if we use "reverse".

The method of sorting is "stable", that is, if two rows have the same value then they will stay the same way round in the sorted table, rather than being swapped over. For example, if we sort this into reverse index order:

Code	Index	Comment
1	1	"Originally row 1"
2	2	"Originally row 2"
2	2	"Originally row 3"
3	3	"Originally row 4"

then we get

Code	Index	Comment
3	3	"Originally row 4"
2	2	"Originally row 2"
2	2	"Originally row 3"
1	1	"Originally row 1"

As a result note that repeating through this sorted table goes through the original rows in order 4, 2, 3, 1; whereas repeating through the original table in reverse order goes through in order 4, 3, 2, 1. (This is all to explain the word "loosely" in the opening sentence of this section.)

Start of Chapter 16: Tables

Back to §16.10. Adding and removing rows

Onward to §16.12. Listed in...

Example 275: ★★ **Jokers Wild** A deck of cards which can be shuffled and dealt from.

---

§16.12. Listed in...

Tables are especially useful for combining a run of basically similar rules in a simple and concise way. The "listed in" condition, as in

```
if the newfound object is an item listed in the Table of Treasures...
```

looks through a given table (here "table of treasures"), in a given column ("item"), to see if a given value is present ("the newfound object"). If this is successful, the row where it was found is automatically chosen; but if not, note that any existing row selection will be lost, so make use of the row only if the test succeeds.

We can similarly use "... listed in ..." in a description used when specifying an action. Thus:

```
After taking an item listed in the Table of Treasures:
    if there is no time entry:
        now the time entry is the time of day;
        increase the score by the value entry;
    say "Taken!"
```

This assumes a table in the following shape:

```
Table of Treasures

Item                Value       Time
Emerald Brooch      5
Sapphire Bracelet   8
Diamond Coronet     10
```

In effect the table has allowed us to combine three very similar rules into one. The time column records the first time at which the item has been picked up, which starts out blank since at the start of play it has never been picked up. This enables us to award the appropriate number of points on the first occasion only.

Start of Chapter 16: Tables

Back to §16.11. Sorting

Onward to §16.13. Topic columns

Example 276: ★★★ **Noisy Cricket** Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

§16.13. Topic columns

When double-quoted matter appears in a column of a table, Inform will normally treat that as text for printing out. The exception is when the column is called "topic", where it is treated as text for comparing against what the player has typed. There is really only one operation allowed with topic columns, the "...listed in..." construction, but fortunately it is the one most often needed.

Let us suppose that the Sybil has a penchant for telling passers-by which is the Greek muse for what. We might write:

```
After asking the Sybil about a topic listed in the Table of Sybil's
Replies, say "The Sybil declaims for a while, the gist being that the
muse in question looks after [muse entry]."
```

We can then provide a simple table giving her responses:

```
Table of Sybil's Replies

Topic                   Muse
"epic poetry"           "Calliope"
"history"               "Clio"
"love poetry"           "Erato"
"music"                 "Euterpe"
"tragedy"               "Melpomene"
"sacred poetry"         "Polyhymnia"
"dancing"               "Terpsichore"
"comedy"                "Thalia"
"astronomy"             "Urania"
```

"tidiness"
"massage"
"hair-cuts"

Topics can use the full range of abilities of the "understanding" system which Inform uses to parse text, and which will be the subject of a later chapter. For now, note that the Sybil's topics might equally include "flora/eve" (matching the single word "flora" or the single word "eve"), or something more elaborate such as:

"Bridget" or "Bridge" or "Bridget Jones"

See **Understand** *for the system Inform uses to parse text*

Start of Chapter 16: Tables

Back to §16.12. Listed in...

Onward to §16.14. Another scoring example

Example 277: **Merlin** A REMEMBER command which accepts any text and looks up a response in a table of recollections.

Example 278: **Questionable Revolutions** An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

Example 279: **The Queen of Sheba** Allowing the player to use question words, and using that information to modify the response given by the other character.

---

§16.14. Another scoring example

```
To record (T - text) as achieved:
    choose row with a citation of T in the Table of Tasks Achieved;
    if there is no time entry:
        now time entry is the time of day;
        increase the score by the points entry.
```

The phrase above expects to see a table like this one:

```
Table of Tasks Achieved

Points  Citation  Time
3       "pride"
3       "anger"
2       "avarice"
4       "envy"
1       "lust"
2       "gluttony"
3       "sloth"
```

The middle column records the tasks to be achieved, the first column records the points on offer for each: the final column, initially blank, will store the times at which the tasks are first achieved.

```
Before eating, record "gluttony" as achieved.
```

The first time we record "gluttony" as achieved, 2 points will be awarded and the time will be logged in the Table, but on all subsequent occasions nothing will happen. So the combination of the phrase and the Table will look after a scoring system based on achieving specific goals (probably not the seven deadly sins, of course). We can, if we choose, use the same system to display a log of recent accomplishments:

```
repeat through the Table of Tasks Achieved in reverse time order:
    say "[time entry]: [citation entry] ([points entry])."
```

Start of Chapter 16: Tables
Back to §16.13. Topic columns

Onward to §16.15. Varying which table to look at

Example 280: ★★★ **Goat-Cheese and Sage Chicken** Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this chapter.

---

§16.15. Varying which table to look at

So far, we have always used fixed table names when referring to tables: for instance in source like "sort the Table of Recent Monarchs in accession order", we refer to the "Table of Recent Monarchs", a definite and explicitly named table.

With a little care, however, we are allowed to have variables which themselves hold the names of tables. This opens up the possibility of more elaborate ways of storing and interconnecting information in table form, but is probably best avoided until it becomes necessary.

For example, suppose we have two different tables with the same basic structure:

Table 1 - Nifty Opening Plays in US Scrabble

word	score
"zjiks"	128

Table 2 - Nifty Opening Plays in UK Scrabble

word	score
"quartzy"	126
"squeezy"	126

We could then record which one of these tables to use in a variable:

```
The lexicon is a table name that varies. The lexicon is Table 1.
```

Note that for this purpose, the kind of value is a special kind called "table name", not "table". (The word "table" already has too many meanings and we must be careful to avoid ambiguities here.) We could make use of this as follows, for instance:

```
To flip tables:
    say "You exchange dictionaries, lexically crossing the Atlantic. ";
    if the lexicon is Table 1, now the lexicon is Table 2;
    otherwise now the lexicon is Table 1;
    choose a random row in the lexicon;
    say "Did you know that according to [the lexicon], [word entry]
scores [score entry]?"
```

which produces text such as

```
You exchange dictionaries, lexically crossing the Atlantic. Did you
know that according to Table 1 - Nifty Opening Plays in US Scrabble,
muzjiks scores 128?
```

Start of Chapter 16: Tables

Back to §16.14. Another scoring example

Onward to §16.16. Defining things with tables

Example 281: ★★ **Farewell**
People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.

§16.16. Defining things with tables

Suppose we need to create a collection of items which differ in their properties, but are basically part of a larger pattern. For instance, here we set up what we need to make a collection of coloured shirts:

```
A jersey is a kind of thing. A jersey is wearable. A jersey has a
number called year established. A jersey has a text called citation.
The description of a jersey is "Since [year established], the Tour de
France has awarded this jersey to the [citation]."
```

Now we have the pattern, but making the actual shirts is tedious and repetitive:

```
The yellow jersey is a jersey. The year established of the yellow
jersey is 1919. The citation of the yellow jersey is "race leader". The
polkadot jersey...
```

And so on. Instead, we can use a table to abbreviate all of this:

```
"Tour des Maillots"
```

```
The Staging Area is a room. A jersey is a kind of thing. A jersey is
wearable. Some jerseys in the Staging Area are defined by the Table
of Honorary Jerseys. The description of a jersey is "Since [year
established], the Tour de France has awarded this jersey to the
[citation]."
```

```
Table of Honorary Jerseys

jersey                  year established    citation
yellow jersey           1919                "race leader"
polkadot jersey         1933                "King of the Mountains"
green jersey            1953                "highest points scorer on sprints"
white jersey            1975                "best young rider aged 25 or less"
```

The first column provides names for the new things to be created. Subsequent columns provide property values. Note that we did not need to say that jerseys have a number called "year established" because Inform is able to infer this from the column heading and the presence of numbers in the column; similarly for "citation". Lastly, note that if any entry is blank (written "--") then that particular property is simply not set for that particular item.

Note that Inform reads articles such as "the" or "a" in the first column just as it would when something is created with any other sentence.

It's even possible to define kinds this way, though it's rare to need to create many kinds at once. (See the worked example "Reliques of Tolti-Aph" at the Inform website. There's no special syntax needed: rather than saying "Some jerseys are defined by..." we would say "Some kinds of jersey are defined by...")

Start of Chapter 16: Tables

Back to §16.15. Varying which table to look at

Onward to §16.17. Defining values with tables

Example 282: **Sweeney** A conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

Example 283: **Introduction to Juggling** Assortment of equipment defined with price and description, in a table.

---

§16.17. Defining values with tables

Just as we can define many similar things (or kinds) using a table, we can also define a whole run of new values. Again, this avoids unnatural prose like

> The chemical elements are Hydrogen, Helium, Lithium, ..., and Ununquadium.

We can give these new values properties, too. For example:

> Solar distance is a kind of value. 1000 AU specifies a solar distance.
> Planet is a kind of value. The planets are defined by the Table of Outer Planets.

```
Table of Outer Planets

planet  semimajor axis
Jupiter  5 AU
Saturn  10 AU
Uranus  19 AU
Neptune  30 AU
Pluto  39 AU
```

creates five values of the kind "planet", but it also makes a property called "semimajor axis" which belongs only to these five values. Thus:

```
say "Pluto orbits at [semimajor axis of Pluto]."
```

produces "Pluto orbits at 39 AU." We can both use and change this value:

```
Praying is an action applying to nothing. Understand "pray" as
praying.
Instead of praying:
    now the semimajor axis of Pluto is 1 AU;
    say "Your prayers are answered, and the Almighty moves Pluto in
closer to the fire."
```

Similar properties would be made for each column of the table after the first (there can be any number of properties, including none). Because the values are created first, before the rest of the table is gone through, we can even use "planet" as one of the values of properties:

```
Table of Outer Planets

planet  semimajor axis  moon assignment
Jupiter  5 AU  Ganymede
Saturn  10 AU  Titan
Uranus  19 AU  Titania
Neptune  30 AU  Triton
Pluto  39 AU  Charon
```

All of this is intended to be closely parallel to defining a whole run of things, such as the coloured jerseys, using a table, but there are two important restrictions: firstly, when a kind of value is defined by table, the table must contain all of its possible values; and secondly, the column names (after the first) cannot coincide with names

of any properties held by any other value (or thing, for that matter). So it is a good idea to give the columns very specific names ("centre of government") rather than vague names which might cause clashes elsewhere ("owner").

Two technical footnotes. In a table used to define a kind of value, blank entries are not left blank: they are filled in with suitable default values. For instance, if the semimajor axis column had been all "--"s except for listing Neptune at "30 AU", say, Inform would deduce that the column was meant to hold a value of kind "solar distance", and would set the solar distances for all of the other planets to be "0 AU". It does this to ensure that "solar distance of P" exists for any planet P.

The second technical note is that we must not sort such a table, because it is used during play to store the properties, and if it were to get rearranged then so would the properties be - with probably disastrous results.

Start of Chapter 16: Tables

Back to §16.16. Defining things with tables

Onward to §16.18. Table continuations

§16.18. Table continuations

A table is an arrangement for putting information together concisely in a single place, so it might seem odd that we sometimes need to divide it up: but once in a while, we do. Suppose we have:

Table of Outer Planets

planet	semimajor axis
Jupiter	5 AU
Saturn	10 AU
Uranus	20 AU
Neptune	30 AU
Pluto	39 AU

But then someone in Chile with a telescope the size of God's own teacup notices something a long, long way out, and the newspapers get terribly excited. We can write an addendum:

Table of Outer Planets (continued)

planet	major axis
Pluto	39 AU
Quaoar	43 AU
Xena	68 AU
Sedna	524 AU

This may seem unnecessary - why not simply add extra rows to the original table? - but it allows us to split the table between different parts of the source text, if we want to, or to continue a table which exists only in an extension. (Thus if we were using an extension which involved the planets, and had a table like this one, we would be able to add new planets without changing the extension.)

The name for the continuation must be identical to the original. The continuation has no existence in its own right: Inform simply splices the two (or more) pieces together, exactly as if the table were all in one piece at the place where it first occurred. Thus the above creates only one table, the "Table of Outer Planets", with nine rows. Each column in the continuation must exist in the original, but not every column need be given: those omitted are filled with blank entries. The columns need not be in the same order. Both original and continuations are allowed to quote a number of blank rows: if so, the combined total is used.

At time of writing the International Astronomical Union has not yet consented to name 2003 UB313 after Xena, the Warrior Princess, but this is surely only a bureaucratic delay. (Footnote: on 24 August 2006 it was demoted to dwarf planet status, like the luckless Pluto, and on 13 September renamed Eris; though its moon's official name, Dysnomia, is an ingenious double-meaning to do with the name of Xena's actress, Lucy Lawless.)

Start of Chapter 16: Tables

Back to §16.17. Defining values with tables

Onward to §16.19. Table amendments

Example 284: ★ **Food Network Interactive** Using a menu system from an extension, but adding our own material to it for this game.

§16.19. Table amendments

Tables can have amendments as well as continuations. The arrangement is much the same: a supplementary table supplies new rows for the original table. But instead of adding the new rows at the end of the original, as a continuation would, an amendment replaces matching rows in the original. (So the original stays the same size.)

The amendment table must have exactly the columns of the original and in the same order. Moreover, each row in the amended table must match exactly one row in the original. For instance:

```
Table of Plans

time    outcome
"10 AM" "take over of Mars"
"1 PM"  "Lake Mead flooded"
"1 AM"  "chocolate bar production doubled"
```

```
Table of Plans (amended)

time    outcome
"1 AM"  "volcanic cave production doubled"
```

creates a three-row Table of Plans, with reference to the chocolate bars struck out.

Amendment rows may be given in any order. The process of matching a row begins at the left-most column: Inform tries to see if any single row in the original table has a matching entry. If none does, a Problem is issued. If more than one do, Inform then looks at the second column, and so on. For instance:

```
Enthusiasm is a kind of value. The enthusiasms are pumped, wired
and languid.

Table of Mental States

feeling     consequence
pumped      "you feel able to run for your life"
pumped      "you feel able to run for President"
wired       "you feel able to run"
languid     "you feel"
```

Table of Mental States (amended)

testing	sequence
pumped	able to run for the Nebraska State Legislature"

Here the amendment is made to the second row of the original table. The value in the leftmost column, "pumped", matches two rows in the original, so Inform moves on to the next column, reads "2", and finds that only one row in the original still qualifies - so that is the one replaced.

For the present, at least, the columns used for matching may only contain: numbers, times, objects, action names, activities, figure names, sound names, truth states and any new kinds of value or units which have been declared.

Start of Chapter 16: Tables

Back to §16.18. Table continuations

Onward to Chapter 17: Understanding: §17.1. Understand

Example 285: ★★ **Trieste** Table amendment to adjust HELP commands provided for the player.

Examples from Chapter 16: Tables

Start of this chapter

Chapter 17: Understanding

Indexes of the examples

★★★ **ExampleDubai** *RB*

An elevator which connects any of 27 floors in a luxury hotel.

The problem of implementing an elevator that opens onto a large number

of floors often challenges novice interactive fiction authors. It also raises a fundamental design problem -- how to implement a large building in an interesting way. It is tempting to write a hotel with an elevator and innumerable tedious and identical floors just for the sake of realism; in many cases it is a better idea simply to omit any locations that contribute nothing to the story or the play of the game.

In charity, though, let us assume that the author has a legitimate reason for wanting to implement an elevator that opens onto some generic floors. We will go whole-hog, and set this in the world's tallest hotel: Burj al-Arab, Dubai.

```
"Dubai"

The Burj al-Arab Lobby is a room. "The 202-suite Burj
al-Arab - or Tower of the Arabs - stands 321 metres
(1,060 feet) high, and floats on its own man-made
island. It is shaped like the sail of a boat; just crossing
the private bridge to reach this place set you back $55."
```

(Since our budget did not run to visiting Burj al-Arab, the descriptions place implicit faith in the hotel's website.)

```
The Assawan Spa is a room. "Treatment rooms,
hydrotherapy baths, oriental massage, stand-up
solarium, sauna, steam rooms and jacuzzi, two
swimming pools, squash court, two fully equipped
fitness studios and an aerobics floor. To the south is a
shopping area, for those who do not find exercise
adequately therapeutic."

The Shopping Area is south of Assawan Spa. "In a
setting that would make a poet sigh, you may enjoy the
services of (among others) Bulgari, Black Pearl Caviar,
Albarajeel Carpet Shop, Abdul Samed Al Qurashi
(amber and Arabic perfumes), Dianoor (jewellery), and
the Commercial Bank of Dubai."

The Al Falak Ballroom is a room. "A palatial, two-tiered,
```

domed ballroom crowned with a unique crystal chandelier."

Sahn Eddar is a room. "At the base of the world's tallest atrium, the Sahn Eddar restaurant offers light fine fare and Afternoon Tea. At the center, a 32-meter water column leaps toward the roof of the atrium."

Al Mahara is a room. "After the elevator, you must take a three-minute virtual submarine voyage to reach this seafood restaurant. A magnificent oval aquarium, full of sharks, is visible from every table."

Al Iwan is a room. "Middle-eastern food in an environment of dramatic black, red, and gold."

Al Muntaha is a room. "A top-floor restaurant with a magnificent view: the name means the ultimate or the highest, suitable for a place that stands at 200 metres above the Arabian Gulf. It offers modern European cuisine; and just off to the south is the Skyview Bar."

The Skyview Bar is south of Al Muntaha. "'A wonderful location for pre- and post dinner drinks,' claims the hotel's brochure, and certainly you can't fault the view."

The Juna Lounge is a room. "Two humidors that offer one of the largest selections of the finest Havana cigars in Dubai."

The Lift is a room. "This is not a mere elevator: it is an express panoramic lift, traveling six meters a second, and capable of taking you from lobby to the rooftop restaurant in an astonishingly short time. The

illuminated number above the door says [current level of the Lift] - though you can make it move by pressing a numbered button."

The Presidential Suite is a room. "Astonishingly, this is not the most elaborate or expensive of suites; there is another, the Royal, upstairs of here."

The Royal Suite is a room. "The brochure described this as 'the last word in luxury,' and you have to admit that it is certainly the last word in [italic type]something[roman type]. A vast carpet of patterned red and gold stretches from you to the sofa; beyond which, in the distance, you make out several bedrooms and bathrooms outfitted in Carrera marble. There is also, of course, a private cinema."

Table of Floors

floor	
0	Al Mahara
1	Burj al-Arab Lobby
2	Al Iwan
3	Juna Lounge
4	Sahn Eddar
15	Al Falak Ballroom
18	Assawan Spa
24	Presidential Suite
25	Royal Suite
27	Al Muntaha

The elevator exterior is a backdrop. It is not scenery. The initial appearance of the elevator exterior is "You can enter the elevator here." It is in Generic Floor, Al Mahara, the Lobby, Al Iwan, Juna Lounge, Sahn Eddar, Al Falak, Assawan Spa, Presidential Suite, Royal Suite, and Al Muntaha.

Before entering the elevator exterior, try going inside instead.

Instead of going inside in the presence of the elevator exterior:
    if there is a level corresponding to a floor of the location in the Table of Floors:
        let the present level be the level corresponding to a floor of the location in the Table of Floors;
        now the current level of the Lift is the present level;
    otherwise:
        now the current level of the Lift is the current level of Generic Floor;
    move the player to the Lift.

The Lift has a number called current level. The current level of the Lift is 1. Instead of going up in the Lift: say "You'll have to select a specific floor; your options range from 0 to 27." Instead of going down in the Lift: try going up instead. The Lobby is outside from the Lift.

Before going outside in the Lift:
    if there is a floor corresponding to a level of the current level of the Lift in the Table of Floors:
        let the other place be the floor corresponding to a level of the current level of the Lift in the Table of Floors;
        move the player to the other place instead;
    otherwise:
        now the current level of the Generic Floor is the current level of the Lift;
        move the player to the Generic Floor instead.

The Generic Floor is a room. The Generic Floor has a number called current level. The printed name of the Generic Floor is "Floor [current level of the Generic Floor]". "A long hallway between suites, some of which run up to $15,000 a night."

Understand "push [number]" as pressing button.

```
Understand "push [number] button" as pressing button.
Understand "push button [number]" as pressing button.
Pressing button is an action applying to one number.


Check pressing button:
    if the player is not in the Lift, say "You cannot control
the express panoramic lift unless you are yourself
inside." instead;
    if the number understood is the current level of the
Lift, say "The lift pings politely and reopens its doors,
since you are already on floor [number understood]."
instead;
    if the number understood is greater than 27, say
"There are only 27 floors." instead;
    if the number understood is less than 0, say "You
cannot go below the ground floor in this elevator."
instead.


Carry out pressing button:
    now the current level of the Lift is the number
understood;
    say "You press button [the number understood]. The
lift whirs into action and moves to the correct level."


Test me with "press 3 / in / press button 3 / look / out / in
/ press 27 / out / s / in / n / in / press 15 / out / in / press
18 / out / s / in / n / in / press 4 button / out".
```

This will all work very well, unless the player has portable objects; in that case, anything he drops on the Generic Floor will be there every time he goes back -- whether it's masquerading as Floor 6 or Floor 23. There are there are two ways round this -- (i) the cheeky way. When we drop something, the unobtrusive yet ever-vigilant maids pick it up and take it down to the Foyer's lost property office; and (ii) the super-duper way, in which things are moved out of play but with their floor numbers remembered, so that the scenario can be reconstructed each time. (i) is probably in fact the more true-to-life, considering the hotel's boasts about its service, but we will demonstrate both methods.

Here is the version with vigilant maids:

```
The player carries a shopping bag. In the bag are a
diamond necklace, a small rug, and a jar of caviar.
```

```
Carry out pressing button:
    now every portable thing enclosed by the Generic
Floor is in the Office.
```

```
The Office is south of the Lobby. "Here the maids collect
everything abandoned by careless guests." The printed
name of the Office is "Lost and Found Office".
```

```
Test maids with "in / press 6 / out / drop all / in / out / in /
press 23 / out / in / press 1 / out / s / get all".
```

Notice that we tie the maid service to the pressing of the lift button, so that if the player just goes into the lift and comes out again, the maids will not have had a chance to clear his possessions.

Alternatively:

```
The player carries a shopping bag. In the bag are an
evening gown, a bolero jacket, and tickets to the Wild
Wadi Animal Park.
```

```
Carry out pressing button when something is in the
Generic Floor:
    repeat with item running through portable things in
the Generic Floor:
        clear the item;
    repeat with item running through portable things
enclosed by the Generic Floor:
        clear the item.
```

The "enclosed by" line clears even things left on, say, small un-portable side-tables or whatever; but because we do "in" first, we make sure to move any containers or supporters undisturbed. The next bit could be more tidily incorporated into our previous "before going outside" rule, but since we are writing this code to be optionally pasted onto the end of the first bit, we'll express the rule separately:

```
Before going outside in the Lift when something is in
Limbo:
    unless there is a floor corresponding to a level of the
current level of the Lift in the Table of Floors:
        repeat with item running through things in Limbo:
            if the current level of the item is the current level
of the Lift, move the item to the Generic Floor.



A thing has a number called current level.



To clear (item - a thing):
    now the current level of the item is the current level of
the Lift;
    move item to Limbo.



Limbo is a room.



Test management with "get all from bag / in / press 22 /
out / drop tickets / in / press 23 / out / drop gown / in /
press 22 / out / get tickets / in / press 23 / out / get
gown".
```

And now we have a situation in which the player's valuables are left untouched wherever in the hotel he happens to abandon them.

Incidentally, this example was almost set in an entirely different location: the largest hotel in the world may some day be the Ryugyong Hotel in Pyongyang, North Korea, with its 105 floors -- but for some years construction halted at the creation of the building's huge concrete shell.

ExamplePort Royal 4 *RB*

A cell window through which the player can see people who were in Port Royal in the current year of game-time.

Our protagonist is imprisoned in Port Royal, waiting out his years, and sometimes through the window of his cell he is able to see someone.

We are, however, obsessive about historical accuracy, so we provide a table of people who really lived in the city, together with the year in which their existence is attested. We want these people to appear in the description only in the year when they are known to have been present. (After all, mortality was high in Port Royal and new people were constantly arriving, so someone's presence one year is no guarantee of their continued existence the next.)

```
"Waiting for Godot, Chyrurgeon"


New New Prison is a room. "You have a not very
interesting room. Through the window you see passing
[current denizen]."


Instead of waiting:
    increment the current year;
    say "It is now the year [the current year].";
    try looking.


When play begins: now the right hand status line is
"[current year]".


Every turn:
    if the current year is 1692:
        say "It turns out you have remained imprisoned
until the great earthquake of 1692! Oops.";
        end the story.


Current year is a number that varies. The current year is
1664.
```

```
To say current denizen:
    repeat through the Table of Occupations and People:
        if the date attested entry is the current year:
            say "[nickname entry] [family entry], [trade
entry]";
            blank out the whole row;
            rule succeeds;
    say "absolutely no one".
```

It is possible to look up a row corresponding to, say, a specific year value using "listed in", but repeat through is convenient here because we know that we will never wind up trying to print entries when no row can be successfully selected.

Table of Occupations and People

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

"[illegible]"
"[illegible]ant"
"[illegible]ez"
"[illegible]"
"[illegible]"
"[illegible]"
"[illegible]"
"[illegible]ker"
"[illegible]surum"
"[illegible]"
"[illegible]"
"[illegible]ker"
"[illegible]ge"
"[illegible]"
"[illegible]"
"[illegible]ker"
"[illegible]lmaster"
"[illegible]ght"
"[illegible]m"
"[illegible]"
"[illegible]"
"[illegible]an"
"[illegible]"
"[illegible]yman"
"[illegible]"
"[illegible]"
"[illegible]keeper"
"[illegible]keeper"
"[illegible]keeper"
"[illegible]keeper"

Test me with "wait / wait / wait / wait / wait / wait / wait / wait / wait / wait".

---

273 **Example If It Hadn't Been For...** *RB*

A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

We start out by giving ourselves a capacious recording device:

"If It Hadn't Been For..."

```
The digital recorder is a device. The description is "A
noise-activated recorder, which time-stamps each
recording segment. It has space for about 60 short
recordings."


Every turn:
    if the digital recorder is switched on and the number
of blank rows in the Table of Recorded Content is 0,
now the recorder is switched off.


Table of Recorded Content

time stamp
sound
action
text
with 60 blank rows.
```

And most of what follows is attaching sounds to various events. (We could have made noises associated with all the actions, but for simplicity we stuck to a few.)

The thing to note here is that the recording happens as part of Carry out, not as part of Report, so sounds will be recorded even when they are the result of non-player action when the player is not even in the room.

```
Carry out opening something in the presence of the
switched on recorder:
    record "A creaking noise, as of something being
opened."


Carry out someone opening something when the
switched on recorder can see the noun:
    record "A creaking noise, as of something being
opened."


Carry out closing something in the presence of the
switched on recorder:
    record "A creaking followed by a slam."
```

```
Carry out someone closing something in the presence
of the switched on recorder:
    record "A creaking followed by a slam."


Carry out someone going to a room (called destination)
in the presence of the switched on recorder:
    if the destination is the holder of the recorder, record
"Footsteps, growing louder.";
    otherwise record "Footsteps, fading out."


Carry out going to a room (called destination) in the
presence of the switched on recorder:
    if the destination is the holder of the recorder, record
"Footsteps, growing louder.";
    otherwise record "Footsteps, fading out."


Carry out someone eating something in the presence of
the switched on recorder:
    record "Loud uncouth chewing sounds."


Carry out eating something in the presence of the
switched on recorder:
    record "Distant muffled chewing sounds."


To record (noise - some text):
    if the number of blank rows in the Table of Recorded
Content is 0, rule succeeds;
    choose a blank row in the Table of Recorded
Content;
    now time stamp entry is the time of day;
    now sound entry is noise.


Understand "play [something]" as listening.
```

```
Instead of listening to the recorder:
    if the number of filled rows in the Table of Recorded
Content is 0, say "The recorder remains blank." instead;
    repeat through the Table of Recorded Content:
        say "[line break][time stamp entry]: [sound entry]";
    say paragraph break.


The Haunted House is a room. The squeaky cupboard
is an openable enterable closed fixed in place container
in the House. The ghost is a man in the cupboard. The
Lawn is outside from the Haunted House.


Instead of opening the closed cupboard when the ghost
is in the cupboard: say "The cupboard stubbornly
refuses to open."


Every turn when the player is not in the House:
    if the ghost is in the cupboard:
        try the ghost exiting;
    otherwise if the cupboard is open:
        try the ghost closing the cupboard.


Before someone exiting when the person asked is in a
closed container (called the trap):
    try the person asked opening the trap.


Before someone entering a closed container: try the
person asked opening the noun.


Before going to the House when the House contains the
ghost:
    try the ghost entering the cupboard;
    try the ghost closing the cupboard.
```

```
The player carries the recorder, chips, and a sandwich.
The sandwich is edible. The chips are edible.


Carry out someone eating the chips in the presence of
the switched on recorder: record "An incredible racket of
a packet being opened." Carry out eating the chips in
the presence of the switched on recorder: record "An
incredible racket of a packet being opened."


Test me with "open cupboard / drop recorder / switch it
on / eat chips / out / wait / wait / wait / in / switch
recorder off / play recorder".
```

Now we're at liberty to record evidence of the ghost getting out of the cupboard and getting back in, while we ourselves stand about on the lawn.

ExampleOdyssey

A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

```
"Odyssey"


Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.


Athena is a woman in Athens.
```

Athena will proceed, unless delayed, through a list of locations stored in a simple table. Rather than using Inform's route-finding abilities ("the best route from..."), we simply move Athena from one location to the next, not even using the going action: she moves in mysterious ways, as befits a goddess.

```
Table of Athena's Movement
destination
Thebes
Delphi
Thebes
Athens
Corinth
Mycenae
```

```
Every turn when Athena is active:
    repeat through the Table of Athena's Movement:
        let last space be the location of Athena;
        if Athena can be seen by the player, say "Athena
heads to [the destination entry].";
        move Athena to destination entry;
        if Athena can be seen by the player, say "Athena
arrives from [the last space].";
        blank out the whole row;
        break.
```

By blanking out the table line by line, we make sure that we never lose our place in the path.

Since we want the player to be able to talk to Athena, we need a way to stall her in her path, as well.

```
Athena can be active or passive. Athena is active.
```

```
Before doing something to Athena:
    now Athena is passive;
    say "Athena waits around patiently, though you can
tell she would like to leave..."
```

```
Instead of telling Athena about something:
    say "She watches you patiently as though to say that
she already knows."

Instead of asking Athena about something:
    say "Her response is inscrutably ancient and Greek.
Afterwards you remember only the flash of bright eyes."
```

Finally, we do need to wake Athena up again if she has become passive. The following rule will occur after the movement rule just because of code ordering, though we could make matters more explicit if we needed to:

```
Every turn when Athena is passive:
    now Athena is active.

Test me with "east / northwest / wait / examine athena /
wait".
```

---

275 **Example** **Jokers Wild** *RB*

A deck of cards which can be shuffled and dealt from.

Suppose we want a deck of cards which the player can shuffle and draw from. Our first (rather tedious) task is merely to set up the deck as a table:

```
"Jokers Wild"

Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades.
```

Table of Cards

value	suit
1	diamonds
2	diamonds
3	diamonds
4	diamonds
5	diamonds
6	diamonds
7	diamonds
8	diamonds
9	diamonds
10	diamonds
11	diamonds
12	diamonds
13	diamonds
1	spades
2	spades
3	spades
4	spades
5	spades
6	spades
7	spades
8	spades
9	spades
10	spades
11	spades
12	spades
13	spades
1	hearts
2	hearts
3	hearts
4	hearts
5	hearts
6	hearts
7	hearts
8	hearts
9	hearts
10	hearts
11	hearts
12	hearts
13	hearts
1	clubs
2	clubs
3	clubs
4	clubs
5	clubs
6	clubs
7	clubs
8	clubs
9	clubs

10 of clubs
11 of clubs
12 of clubs
13 of clubs

We're going to describe the higher numbers as face cards, so it helps to write a new "to say" phrase.

```
To say (count - a number) as a card value:
    choose row with a value of count in the Table of
Value Names;
    say "[term entry]".
```

```
Table of Value Names

term value
1 "ace"
2 "deuce"
3 "three"
4 "four"
5 "five"
6 "six"
7 "seven"
8 "eight"
9 "nine"
10 "ten"
11 "jack"
12 "queen"
13 "king"
```

Now we get the shuffling of the deck from "sort in random order", so:

```
Understand "shuffle" as shuffling. Shuffling is an action
applying to nothing.
```

```
Carry out shuffling:
    sort the Table of Cards in random order;
    say "You expertly rearrange the cards.".
```

```
When play begins:
    sort the Table of Cards in random order.
```

This will continue to work properly even as the deck is partially depleted. Speaking of which, suppose we want the player to be able to toss the cards one-by-one into a hat. They are going to need to be removed from the deck, so:

```
Understand "toss" or "toss a card" or "toss card" as
tossing.


Tossing is an action applying to nothing.


Check tossing:
    if the number of filled rows in the Table of Cards is 0,
say "The deck is empty." instead.


Carry out tossing:
    repeat through the Table of Cards:
        let new value be value entry;
        let new suit be suit entry;
        say "You throw the [value entry as a card value] of
[suit entry] at the top hat, and [if a random chance of 1
in 3 succeeds]hit[otherwise]miss[end if].";
        blank out the whole row;
        rule succeeds.
```

If we wanted to simulate a slightly more stimulating game, we could instead have a second table to represent the player's hand of cards and record each card drawn. That would get long for the purposes of example, however, so instead we will just admit that the player's life is an empty husk of existence:

```
The Empty Room is a room. "It has come to this: sitting
on the bare floor of Lulu's apartment with nothing to
amuse you but a deck of cards and the top hat from last
```

year's act. You reckon [the number of filled rows in the Table of Cards in words] cardtosses are all that stand between you and the utter pointlessness of existence.

Once again you curse Lulu for running off with that joker."

The player is carrying the deck of cards. The top hat is an open container in the Empty Room. It is scenery.

Test me with "toss / again / again / again / again / again / again / again".

---

276 **ExampleNoisy Cricket** *RB*

Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

Our previous experiments into liquid have not dealt with the possibility of mixing components, but that is because for most games, tracking the details of mixture is overkill.

But let's suppose that this time we do want to have mixed liquids; moreover, we want a way to describe the mixtures to the player inventively, so that if he hits specific combinations those combinations are recognized: calling the result a martini, say, rather than just "a mixture of vodka and vermouth".

The implementation that follows relies on a fairly simple idea from linear algebra. Any given liquid can be expressed as a vector in N-space, where N is the number of available ingredients and the length of the vector depends on how much of each ingredient is used; then we find the recipe that best describes the liquid by taking the dot product of our liquid vector with a bunch of sample vectors and selecting the one with the largest result.

If this does not make sense, don't worry: it's not necessary to understand the idea to use the code.

Any implementation involving a large number of place values is always a bit challenging in integer arithmetic. This examples assumes that no bodies of liquid will ever be very large, and that the proportions of ingredients in a mixture will not be vastly askew. (No 20-parts-to-1 proportions, for instance.) This probably works reasonably well for the cocktails that we make the basis of the example.

"Noisy Cricket"

Part 1 - Volumes and Mixtures

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called creme de menthe volume. A fluid container has a volume called vodka volume. A fluid container has a volume called cacao volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The creme de menthe volume of a fluid container is usually 0.0 fl oz. The vodka volume of a fluid container is usually 0.0 fl oz. The cacao volume of fluid container is usually 0.0 fl oz.

To decide what volume is the current volume of (item - a fluid container):
    let total be the creme de menthe volume of the item;
    increase total by the vodka volume of the item;
    increase total by the cacao volume of the item;
    decide on total.

```
Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the nominal descriptor
of the noun] at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [adjectival descriptor of the
noun] [nominal descriptor of the noun]."
```

```
Adjectival descriptor is a kind of value. The adjectival
descriptors are strong, chocolatey, minty, perfect, and
pure.
```

```
Nominal descriptor is a kind of value. The Nominal
descriptors are creme de menthe, vodka, creme de
cacao, grasshopper, chocolate vodka, mint vodka,
chocolate martini, mintini, chocolate mint martini.
```

Our table of mixtures is expressed in parts: so if a recipe contains one part X and two parts Y, we would put "1" in the first column and "2" in the second column.

```
Table of Mixtures

[illegible]comp
1 minty creme de menthe
1 strong vodka
1 chocolatey creme de cacao
1 strong vodka
1 chocolatey grasshopper
1 chocolatey martini
1 chocolatey vodka
1 strong vodka
1 chocolatey mint martini
```

```
A fluid container has an adjectival descriptor. A fluid
container has a nominal descriptor. Understand the
adjectival descriptor property as describing a fluid
container. Understand the nominal descriptor property
as describing a fluid container.
```

```
To decide what number is (quantity - a number)
squared:
    decide on quantity times quantity.



To score mixtures in (item - a fluid container):
    repeat through Table of Mixtures:
        let total line parts be creme de menthe comp entry
squared;
        let total line parts be total line parts plus vodka
comp entry squared;
        let total line parts be total line parts plus cacao
comp entry squared;
        let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the
vodka volume of item;
        let cacao score be cacao comp entry times the
cacao volume of item;
        let total score be creme de menthe score plus
vodka score;
        let total score be total score plus cacao score;
        let total score be total score times calibration for
total line parts;
        now rating entry is total score;
        if total line parts is 1, now adjectival descriptor
entry is pure;
        otherwise now adjectival descriptor entry is perfect;
        [and for creme de menthe...]
        now creme de menthe comp entry is creme de
menthe comp entry plus 1;
        let total line parts be creme de menthe comp entry
squared plus vodka comp entry squared;
        let total line parts be total line parts plus cacao
comp entry squared;
        let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the
vodka volume of item;
        let cacao score be cacao comp entry times the
cacao volume of item;
        let total score be creme de menthe score plus
vodka score;
        let total score be total score plus cacao score;
        let total score be total score times calibration for
total line parts;
        if total score is greater than rating entry, now
```

```
adjectival descriptor entry is minty;
        now creme de menthe comp entry is creme de
menthe comp entry minus 1;
        [and for vodka...]
        now vodka comp entry is vodka comp entry plus 1;
        let total line parts be creme de menthe comp entry
squared plus vodka comp entry squared;
        let total line parts be total line parts plus cacao
comp entry squared;
        let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the
vodka volume of item;
        let cacao score be cacao comp entry times the
cacao volume of item;
        let total score be creme de menthe score plus
vodka score;
        let total score be total score plus cacao score;
        let total score be total score times calibration for
total line parts;
        if total score is greater than rating entry, now
adjectival descriptor entry is strong;
        now vodka comp entry is vodka comp entry minus
1;
        [and for cacao...]
        now cacao comp entry is cacao comp entry plus 1;
        let total line parts be creme de menthe comp entry
squared plus vodka comp entry squared;
        let total line parts be total line parts plus cacao
comp entry squared;
        let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the
vodka volume of item;
        let cacao score be cacao comp entry times the
cacao volume of item;
        let total score be creme de menthe score plus
vodka score;
        let total score be total score plus cacao score;
        let total score be total score times calibration for
total line parts;
        if total score is greater than rating entry, now
adjectival descriptor entry is chocolatey;
        now cacao comp entry is cacao comp entry minus
1.



To identify mixture in (item - a fluid container):
    score mixtures in item;
    sort Table of Mixtures in reverse rating order;
```

```
    choose row 1 in Table of Mixtures;
    now nominal descriptor of the item is nominal
descriptor entry;
    let sample vodka be vodka comp entry; [Now keep
track of all these]
    let sample creme de menthe be creme de menthe
comp entry;
    let sample cacao be cacao comp entry;
    if rating entry divided by 100 is the current volume of
the item:
        now adjectival descriptor of the item is pure;
    otherwise:
        now adjectival descriptor of the item is adjectival
descriptor entry.



To decide what number is the raw quantity of (item
volume - a volume):
    let raw be item volume divided by 0.5 fl oz;
    decide on raw.



To decide what number is calibration for (total - a
number):
    if total is an initial listed in the table of Multipliers,
decide on result entry;
    decide on 21.
```

Here we cheat on our arithmetic. The following chart just provides values corresponding roughly to 1/(sqrt (x)), but since Inform does not deal very gracefully with square roots or fractions, we will calculate this elsewhere and just supply the answers in the code:

```
Table of Multipliers

initial   result
1         100
2         71
3         57
4         50
5         44
6         41
7         38
8         35
9         33
10        31
```

```
descriptor of the target]";
        omit contents in listing.


Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."


Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.


Understand "drink from [fluid container]" as drinking.


Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [nominal descriptor of the
noun] within." instead;
    otherwise:
        let cacao loss be the consumed cacao of the noun
out of sip volume;
        let creme de menthe loss be the consumed creme
de menthe of the noun out of sip volume;
        let vodka loss be the consumed vodka of the noun
out of sip volume;
        decrease the cacao volume of the noun by the
cacao loss;
        decrease the creme de menthe volume of the noun
by creme de menthe loss;
        decrease the vodka volume of the noun by vodka
loss;
        say "You take a sip of [the nominal descriptor of
the noun][if the noun is empty], leaving [the noun]
empty[end if].".


Sip volume is a volume that varies. Sip volume is 0.5 fl
oz.
```

To decide what volume is the consumed cacao of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the cacao volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;
    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the cacao volume of the item, decide on the cacao volume of the item;
    decide on consumed volume.

To decide what volume is the consumed creme de menthe of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the creme de menthe volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;
    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the creme de menthe volume of the item, decide on the creme de menthe volume of the item;
    decide on consumed volume.

To decide what volume is the consumed vodka of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the vodka volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;
    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the vodka volume of the item, decide on the vodka volume of the item;
    decide on consumed volume.

Part 2 - Filling

Understand the command "fill" as something new.

Understand "fill [something] with/from [something]" as filling it with.

Filling it with is an action applying to two things. Carry out filling it with: try pouring the second noun into the noun instead.

Understand "pour [fluid container] in/into/on/onto [fluid container]" as pouring it into. Understand "empty [fluid container] into [fluid container]" as pouring it into.

Understand "pour [something] in/into/on/onto [something]" as pouring it into. Understand "empty [something] into [something]" as pouring it into.

Pouring it into is an action applying to two things.

Check pouring it into:
    if the noun is not a fluid container, say "You can't pour [the noun]." instead;
    if the second noun is not a fluid container, say "You can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly pour [the noun] into itself." instead;
    if the noun is empty, say "No more [nominal descriptor of the noun] remains in [the noun]." instead;
    if the second noun is full, say "[The second noun] cannot contain any more than it already holds." instead.

```
Carry out pouring it into:
    let available capacity be the fluid capacity of the second noun minus the current volume of the second noun;
    if the available capacity is greater than the current volume of the noun, now the available capacity is the current volume of the noun;
    let cacao loss be the consumed cacao of the noun out of available capacity;
    let creme de menthe loss be the consumed creme de menthe of the noun out of available capacity;
    let vodka loss be the consumed vodka of the noun out of available capacity;
    decrease the cacao volume of the noun by the cacao loss;
    decrease the creme de menthe volume of the noun by creme de menthe loss;
    decrease the vodka volume of the noun by vodka loss;
    increase the cacao volume of the second noun by the cacao loss;
    increase the creme de menthe volume of the second noun by creme de menthe loss;
    increase the vodka volume of the second noun by vodka loss.



Report pouring it into:
    identify mixture in noun;
    identify mixture in second noun;
    say "[if the noun is empty][The noun] is now empty; [otherwise][The noun] now contains [current volume of the noun in rough terms] of [nominal descriptor of the noun]; [end if]";
    say "[the second noun] contains [current volume of the second noun in rough terms] of [adjectival descriptor of the second noun] [nominal descriptor of the second noun][if the second noun is full], and is now full[end if]."



Understand "of" as a fluid container.



Part 3 - Scenario
```

When play begins: say "When you decided to try Mixology WS102 (*cross-listed with Women's Studies), you envisioned yourself writing essays about gender discrimination during the Prohibition, say, or reading essays on male vs. female metabolism of alcohol. But no: MxWS102 turns out to be about... mixing the perfect chocolate mint martini."

The College of Mixology is a room. The bar is a supporter in the college.

The cocktail glass is a fluid container carried by the player. The fluid capacity of the cocktail glass is 4.0 fl oz.

The flask is a fluid container carried by the player. The vodka volume of the flask is 4.0 fl oz.

The jigger is a fluid container carried by the player. The fluid capacity of the jigger is 1.0 fl oz.

The small measure is a fluid container carried by the player. The fluid capacity of the small measure is 0.5 fl oz.

The decanter is a fluid container on the bar. The fluid capacity of the decanter is 32.0 fl oz. The creme de menthe volume of the decanter is 20.0 fl oz.

The bottle is a fluid container carried by the player. The cacao volume of the bottle is 10.0 fl oz.

Test me with "i / pour flask in jigger / pour jigger in glass / pour bottle in jigger / pour jigger in glass / pour bottle in jigger / pour jigger in glass / pour decanter in jigger / pour jigger in glass / drink glass / g / g / x glass / pour flask in glass".

---

27 ExampleMerlin *RB*

A REMEMBER command which accepts any text and looks up a response in a table of recollections.

"Merlin"

Understand "remember [text]" as remembering.

Remembering is an action applying to one topic.

Carry out remembering:
    say "Nothing comes to mind.".

Instead of remembering a topic listed in the Table of Recollections:
    say "[response entry][paragraph break]".

Table of Recollections

Topic	Response
"weather"	"You wish the weather were worse, but not often: it's falling so hard now that the tin rattles and the runoff, on the low side of the roof, would be a tenable source of hydroelectric power."
"hydroelectric power"	"It does not have enough power to operate even a single light bulb in here to turn on, even if you could power it."

"Light bulbs, like so much else, are a thing of your past. Or is it your future? Tricky, the way the world loops round on itself."

"Living backwards has its drawbacks. A tendency to confuse and annoy your friends, being one; the total inability to maintain a stable relationship; and a deep dissatisfaction with most of the bodily processes people enjoy, since they ultimately make you hungrier, colder, or-- no point dwelling on it, really."

"It's not exactly backwards, now is it? It's more like a series of forwardses stuck back to back. As though someone had taken each track of a CD and put them in the exactly wrong order. You miss that. The music on demand."

The Inadequate Shelter is a room. "A piece of corrugated tin, leaned on two sticks, and pathetically augmented with a tire (on one side) and a cardboard box (on the side towards the wind). And that's what you've got between you and the driving rain.

At the moment rain is all you can remember, in fact."

Test me with "remember rain / remember power / remember light bulb / remember future / remember backwards".

---

278 **Example** **Questionable Revolutions** *RB*

An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

"Questionable Revolutions"

Interrogative is a kind of value. The interrogatives are who, what, when, where, how, and why.

Current question is an interrogative that varies.

After asking someone about something: respond to the question. After answering someone that something: respond to the question.

After telling someone about something: say "You're here to ask questions."

Country is a kind of value. The countries are Czechoslovakia, Georgia, Sweden, Italy, Spain.

Table of Information

topic	place	date	definition
"velvet revolution"	Czechoslovakia	1989	"Bloodless revolution in Czechoslovakia, in which popular protests led to the resignation of the communist president Gustav Husak, and the election of Vaclav Havel in his place."
"rose revolution"	Georgia	2003	"Revolution in which President Eduard Shevardnadze was interrupted by protesters in the middle of his speech, and forced to flee."
"spanish revolution"	Spain	1936	"Anarchist and socialist movement during the Spanish civil war."

After reading a command:
    if the player's command includes "[interrogative]",
now the current question is the interrogative understood.

To respond to the question:
    repeat through the Table of Information:
        if the topic understood includes topic entry:
            if the current question is what or the current
question is who, say definition entry appropriately;

        if the current question is when, say date entry appropriately;
        if the current question is where, say place entry appropriately;
        rule succeeds;
    say "[The noun] shrugs."

Understand "ask [someone] [text]" as asking it about.

Comprehension is a kind of value. The comprehensions are vague, erroneous, and correct.

Table of Understanding

[illegible]	[illegible] comprehension
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

When play begins:
    say "Here you are in the first class cabin, but no matter how fancy the seats are, you can still get bored circling over Zurich for three hours on end. To kill time, you and the other passengers are playing a trivia game, and the final topic is your specialty: revolutions."

First Class Cabin is a room. Dr Tweedy is a man in First Class. Ms Finch and Ms Clarion are women in First Class.

To say (year - a number) appropriately:
    choose row with character of the noun in the Table of Understanding;
    if years entry is correct:
        say "'[year],' replies [the noun] promptly.";

```
        increment the quiz score of the noun;
    if years entry is erroneous:
        let guess be a random number between 1900 and
2005;
        say "'[guess]?' guesses [the noun], with an air of
diffidence[if guess is the year]. Which is right, as it
happens[end if].";
        if guess is the year, increment the quiz score of the
noun;
    if years entry is vague:
        let offset be a random number between -5 and 5;
        let year be year + offset;
        say "'I think [year]. About then. Close, anyway,'
replies [the noun][if the offset is 0], getting it right[end
if].";
        if offset is 0, increment the quiz score of the noun.



To say (spot - a country) appropriately:
    choose row with character of the noun in the Table of
Understanding;
    if geography entry is correct:
        say "'[spot],' replies [the noun] promptly.";
        increment the quiz score of the noun;
    if geography entry is erroneous:
        let guess be a random country;
        say "'Er... [guess]?' says [the noun][if guess is the
spot]. Which is of course correct[end if].";
        if guess is the spot, increment the quiz score of the
noun;
    if geography entry is vague, say "'Europe,' replies [the
noun] with confidence."



To say (explanation - some text) appropriately:
    choose row with character of the noun in the Table of
Understanding;
    if general comprehension entry is correct:
        say "'[explanation]'[paragraph break]";
        increment the quiz score of the noun;
    otherwise:
        choose a random row in the Table of Information;
        say "'[definition entry]'[paragraph break]";
        if the definition entry is explanation:
            say "[A random other person who is not the
noun] looks surprised that this came out right.";
            increment the quiz score of the noun.
```

```
A person has a number called quiz score. The quiz
score of Tweedy is 48. The quiz score of Finch is 2. The
quiz score of Clarion is 4.


Definition: a person is other if it is not the player.


When play begins:
    now left hand status line is "T: [quiz score of Tweedy]
F: [quiz score of Finch] C: [quiz score of Clarion]";
    now right hand status line is "[time of day]".


Test me with "dr tweedy, where was the velvet
revolution located / ms finch, when was the rose
revolution / ms finch, what was the rose revolution / ms
clarion, when was the spanish revolution / g / g / ms
finch, when was the spanish revolution".
```

We have so far seen several ways to write conversational characters in Inform, and we will see more before the end of the manual. This naturally raises the question, which should we use? To which the answer is: it depends on the sort of game we're writing, and what we want our characters to do. The more rich and complex the system, the more likely that it will require a lot of content; if we add question types as well as keywords, for instance, we instantly multiply the number of responses we have to write by five or six. It is not worth doing this unless there is some corresponding advantage within the game.

Example **The Queen of Sheba** *RB*

Allowing the player to use question words, and using that information to modify the response given by the other character.

Suppose we want the player to ask questions of slightly more complexity - we might want to build in a system that understood "who", "what", "where", and "when", for instance. We could use a topic table for this, too:

"The Queen of Sheba"

Interrogative is a kind of value. The interrogatives are who, what, when, where, how, and why.

Current question is an interrogative that varies.

After asking someone about something: respond to the question. After answering someone that something: respond to the question.

After telling someone about something: say "You're here to ask questions and test Solomon's wisdom, not to give him a sample of your own."

Table of Wise Answers

question type	reply
"What's the matter?"	"'What vexes the birds is the wind,' Solomon replies. 'The air beaten by their wings becomes agitated, as when a river is stirred and the mud churns up.'"
"What's the matter?"	"'What vexes the clouds is the sun,' replies Solomon."
"What's the matter?"	"'Hungry, are you getting hungry?' he says, and rings a bell to summon servants."
"Why we need to"	"'We need food in order that they must farm and cook and dine together,' Solomon replies. 'Otherwise, they might live apart, each sufficient in himself. But no man can feed himself alone all through his life.'"
"Solomon, he/himself"	"Solomon, he/himself," holding out his arms to each side."
"Solomon, he/himself"	"Solomon, he/himself," he answers."

One of the nice things about this system is that it only resets the "current question" when we get a new question word. For instance, this test will produce different replies to the question about Solomon himself, because

the second time he is still in the mode of answering "what" questions:

```
Test me with "ask solomon about himself / ask solomon
what rain is / ask solomon about himself".
```

If Solomon is to live up to his reputation at all, his wisdom table will have to be quite a bit longer - though one also would want to be careful, because forcing the game to cycle through a really immense table could be quite time-consuming. In fact, for the sake of this example, let's reward the player for managing to stay within the (narrow) range of Solomon's knowledge:

```
The Hall of Almug Tree Pillars is a room. "The pillars of
the room are made of almug tree, the ceiling made of
silk and the floor of glass." Solomon is a man in the Hall
of Almug Tree Pillars. Solomon has a number called
wisdom. The wisdom of Solomon is 0.


Every turn:
    if the wisdom of Solomon is 3:
        say "Truly, Solomon has answered all your
questions, and his wisdom is even as great as you had
heard!";
        end the story saying "Your heart beats strangely
fast".


When play begins, say "'Oh, you've arrived,' says
Solomon."
```

In a real game we'd need to be a great deal subtler. All the same, if we have a character of quite limited resources to present to the player, it's a good idea to give the player some incentive to stay on topic, ask questions the character can answer, and generally interact within the parameters we're prepared for.

Now, this last bit requires some trickery from later chapters, particularly those on Understanding and Activities, to pull the question words out of the player's command:

```
After reading a command:
    if the player's command includes "[interrogative]",
now the current question is the interrogative understood.



To respond to the question:
    repeat through the Table of Wise Answers:
        if the topic understood includes topic entry:
            if the current question is the question type entry:
                say "[reply entry][paragraph break]";
                increment the wisdom of Solomon;
                rule succeeds;
    say "Solomon looks blank, appalled by a question for
which he was not prepared.";
    end the story saying "You have befuddled Solomon!"



Understand "ask [someone] [text]" as asking it about.
```

And now we have a game that will accept (though not always respond very sensibly to) questions of almost any form we might put to another character: ASK SOLOMON WHAT RAIN IS will be answered, but then again, it won't be distinguished from, say, ASK SOLOMON WHETHER THIS PERSISTENT RAIN IS A DIVINE PUNISHMENT OR WHAT.

All the same, a system that allowed the player a bit more specification of questions than simple keyword-use might be useful in a mystery game, for instance, where we might want to let our detective conduct inquiries into specific details. An alternative approach to the rather free one above would be to force the player to use only questions of the form WHAT IS RAIN? or WHO ARE YOU?: this would cut down on false-positive matches. But we might still choose to store the responses in a table of this type.

Example Goat-Cheese and Sage Chicken *RB*

Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this chapter.

Some games provide a FULL SCORE command that gives more information about the player's achievements than SCORE alone. Supposing we wanted to include a FULL SCORE in our game that gave

the kind of score reading described in this chapter:

"Goat-Cheese and Sage Chicken"

Use scoring.

The story headline is "An interactive recipe"

Table of Tasks Achieved

Time	Points	Citation
3 minutes	3	"sauteeing onions"
3	3	"reconstituting apricots"
1	1	"flattening chicken"
1	1	"unwrapping goat cheese"

To record (T - text) as achieved:
    choose row with a citation of T in the Table of Tasks Achieved;
    if there is no time entry:
        now time entry is the time of day;
        increase the score by the points entry.

Requesting the full score is an action out of world.
Understand "full" or "full score" as requesting the full score.

Carry out requesting the full score:
    if the score is 0, say "You have achieved nothing towards supper." instead;
    repeat through the Table of Tasks Achieved in reverse time order:
        say "[time entry]: [citation entry] ([points entry])."

Table of Rankings

Score	Rank
0	"Rank Amateur"
2	"would-be Bobby Flay"
5	"Alton Brown"
8	"Julia Child"

The Kitchen is a room. The description of the Kitchen is "Equipped with many familiar friends: refrigerator, stove, oven; countertop; cabinet for pans and bowls, and a drawer for your tools."

The stove is scenery in the kitchen. It is a supporter. The oven is a container. It is part of the stove. It is closed and openable. The stove's switch is a device. It is switched on. It is part of the stove. The oven's dial is a device. It is switched off. It is part of the oven.

A thing can be heatproof.

Instead of putting something which is not heatproof on the stove when the stove's switch is switched on:
    say "You catch yourself just at the last minute: not a good idea to put [the noun] directly on the stove while it's turned on."

Instead of switching on the stove, try switching on the stove's switch. Instead of switching off the stove, try switching off the stove's switch. Instead of switching on the oven, try switching on the oven's dial. Instead of switching off the oven, try switching off the oven's dial.

Before switching on the oven's dial when the oven is open:
    say "(closing the oven so that it will heat properly) [command clarification break]";
    try closing the oven.

The frying pan is a heatproof unopenable open container on the stove.

The cabinet is a closed openable container in the kitchen. It is scenery. It contains an open unopenable container called a mixing bowl. It contains a portable supporter called a platter. An open unopenable heatproof container called a Calphalon baking dish is in the cabinet. The baking dish has the description "One of those marvelous pieces of kitchen equipment which goes on the stove or in the oven, as you will. The chief thing is never ever to touch it when it is hot, since the handles are metal and the heat retention excellent."

The counter is a supporter in the kitchen. It is scenery. The kettle is a heatproof openable closed container on the counter. Some water is in the kettle.

The water can be cool, warm, or boiling. The printed name of the water is "[water condition] water".

The refrigerator is a closed openable container in the kitchen. It is scenery. Understand "fridge" as the refrigerator.

An ingredient is a kind of thing.

Some onions, some apricots, and some sage are

ingredients on the counter. A chicken breast, an egg, and goat cheese are ingredients in the refrigerator.

The goat cheese can be wrapped, snipped open, or unwrapped. The printed name of the goat cheese is "[goat cheese condition] goat cheese".

The sage can be unwashed, clean, or julienned. The sage is unwashed. The printed name of the sage is "[sage condition] sage".

The apricots can be dried, reconstituted, or chopped. The apricots are dried. The printed name of the apricots is "[apricots condition] apricots".

The chicken breast can be whole, flattened, stuffed, rolled, coated, browned, or baked. The printed name of the chicken breast is "[chicken breast condition] chicken breast".

The onions can be unpeeled, peeled, diced, sauteed, or burnt. [The printed name of the onions is "[onions condition] onions".]

The can of chicken broth is a closed container on the counter. The bottle of white cooking wine and the bottle of Thurston Wolfe PGV are a closed containers in the refrigerator.

The description of the Thurston Wolfe is "A Washington State Pinot Gris-Viognier, 2003. It is said to have 'peach aromas', and, startlingly, the untutored person can detect these without resorting to fantasy.

(It is also supposed to possess a delicate perfume and a moderate body; the label author at least stopped short of 'good sense of humor and likes long walks on the beach')."

Understand the commands "wash" and "rinse" as "clean".

Instead of rubbing the unwashed sage:
    now the sage is clean;
    say "You rinse off the sage. There -- ready to slice."

Instead of cutting the sage:
    say "You'd need to have a knife in hand, first."

Instead of cutting the clean sage when the player is carrying the butcher knife:
    now the sage is julienned;
    say "You slice the sage into thin strips."

Instead of cutting the unwashed sage:
    say "It came from the garden, so it won't have any strange chemicals on it, but you should still give it a rinse for dirt and bugs and so on before using it."

Instead of doing something other than examining or rubbing with the unwashed sage:
    say "It needs to be washed off."

Understand "peel [something]" as peeling.

Peeling is an action applying to one thing.

Instead of peeling the unpeeled onions:
    now the onions are peeled;
    say "You tear away the shining outer skin of the onions, leaving them pale and nekkid. Poor things."

Instead of cutting the diced onions:
    say "That seems unnecessary now."

Instead of cutting the sauteed onions:
    say "Too late; you're well past that stage."

Instead of cutting the burnt onions:
    say "There's no rescuing 'em -- the carbon isn't going to flake off, you know."

Instead of cutting the unpeeled onions:
    say "It would help to peel them first."

Instead of cutting the peeled onions:
    say "You'd need to have a knife in hand, first."

Instead of cutting the peeled onions when the player is carrying the butcher knife:
    now the onions are diced;
    say "You dice the onions neatly. Your own skill brings tears to your eyes."

Instead of opening the goat cheese:
    try peeling the goat cheese instead.

Instead of peeling the unwrapped goat cheese:
    say "The goat cheese is already unwrapped. (Stay focused, stay focused...)"

Before peeling the wrapped goat cheese when the shears are held by the player:
    try cutting the goat cheese.

Instead of peeling the snipped open goat cheese:
    now the goat cheese is unwrapped;
    record "unwrapping goat cheese" as achieved;
    say "Ah, success. The goat cheese is now free of its packet."

Instead of peeling the wrapped goat cheese:
    say "It would help to have a pair of scissors or something -- the packet resists being torn."

Instead of cutting the goat cheese:
    say "No need, at this point."

Before cutting the wrapped goat cheese when the shears are not held by the player and the shears are visible:
    say "(first picking up the shears)[command clarification break]";
    try taking the shears.

Instead of cutting the wrapped goat cheese:
    say "Something to cut with would be useful."

Instead of cutting the wrapped goat cheese when the shears are held by the player:
    now the goat cheese is snipped open;
    say "You neatly snip through the packaging with the shears."

Instead of examining the whole chicken breast:
    say "It is still entire and has yet to be pounded flat."

Instead of examining the flattened chicken breast:
    say "It has been hammered to a thickness of about a half inch. (The recipe said a quarter inch but you're pretty sure it was joking. You have never been able to achieve a quarter inch.)"

Instead of attacking the whole chicken breast:
    say "You need something heavy enough to flatten it with."

Instead of attacking the whole chicken breast when the player is holding the wooden mallet:
    now the chicken breast is flattened;
    record "flattening chicken" as achieved;
    say "You hammer away at the chicken breast, turning all your aggressions into culinary goodness. Several minutes pass. When you are done you have a broad flat chickeny pancake suitable for wrapping about a stuffing."

Before printing the name of onions:
    say "[onions condition] ".

The drawer is an openable closed container. It is part of the counter.

A tool is a kind of thing. A spatula, a spoon, a wooden mallet, some shears, and a ball of twine are tools in the drawer. A butcher knife is a tool carried by the player. Understand "scissors" as the shears.

Instead of burning something:
    say "You'll have to do that the hard way."

Some steam is fixed in place. "Dense clouds of steam fill the room."

Some smoke is fixed in place. "Smoke is beginning to collect near the ceiling."

Sauteeing Onions is a scene. Sauteeing Onions begins when the diced onions are in a hot container.

Definition: a container is hot if it is on the stove and the stove's switch is switched on.

Instead of touching the hot pan:
    say "Ow!"

Scorching Onions is a scene.

Preheating the Oven is a scene. Preheating the Oven begins when the oven's dial is heating.

Definition: a oven's dial is heating if the oven's dial has been switched on for exactly one turn.

Preheating the Oven ends when the time since Preheating the Oven began is five minutes.

When Preheating the Oven begins:
    say "The oven begins to warm up."

When Preheating the Oven ends:
    say "The oven beeps to inform you that it has reached the desired hotness."

Every turn during Sauteeing Onions:
    say "The onions sizzle in the pan."

Every turn during Scorching Onions:
    say "The onions are past their prime and are getting blacker by the moment."

Every turn during Hearing the Kettle Whistle:
    say "The kettle continues to whistle."

Instead of listening to during Hearing the Kettle Whistle:
    say "The only thing you can really hear just at the moment is the kettle."

Instead of smelling the Kitchen during Sauteeing Onions:
    try smelling the onions.

Instead of smelling the onions during Sauteeing Onions:
    say "The onions smell marvelous."

Instead of opening the oven during Preheating the Oven:
    say "It'll never heat if you open it up while it's warming."

Heating Kettle is a scene. Heating Kettle begins when the hot kettle contains cool water.

Before printing the name of the kettle when the kettle is hot:
    say "hot "

When Heating Kettle begins:
    say "The kettle begins to heat up."

Heating Kettle ends when the time since Heating Kettle began is 7 minutes.

Hearing the Kettle Whistle is a scene. Hearing the Kettle Whistle begins when Heating Kettle ends. Hearing the Kettle Whistle ends when the kettle is not hot.

When Hearing the Kettle Whistle begins:
    now the water is boiling;
    say "The kettle begins to burble and whistle shrilly."

```
When Hearing the Kettle Whistle ends:
    say "The kettle's screaming dies off."


Idling is a scene. Idling begins when play begins. Idling
ends when Sauteeing Onions begins.


Sauteeing Onions ends in disaster when Scorching
Onions begins.


Sauteeing Onions ends in success when the onions are
sauteed and onions are not in a hot container.


Definition: a thing is alone if it is in a container which
contains exactly one thing.


Sauteeing Onions ends in mixture when the sauteed
onions are not alone.


When Sauteeing Onions ends in mixture:
    say "The mixture of things in [the holder of the
onions] stops them cooking quite so fast."


When Sauteeing Onions ends in success:
    say "Nice work with the onions."


Every turn:
    if diced onions have been in a hot pan for ten turns:
        say "The onions are starting to look ready.";
        now the onions are sauteed.
```

Scorching Onions begins when Sauteeing Onions ends in disaster. Scorching Onions begins when the alone sauteed onions are in a hot container.

Scorching Onions ends horribly when the time since Scorching Onions began is three minutes. Scorching Onions ends in reprieve when the sauteed onions are not in a hot container. Scorching Onions ends in mixture when the sauteed onions are not alone.

When Scorching Onions ends in mixture:
    record "sauteeing onions" as achieved;
    say "The mixture of things in [the holder of the onions] stops them cooking quite so fast."

When Scorching Onions ends horribly:
    move smoke to Kitchen;
    now the onions are burnt.

When Scorching Onions ends in reprieve:
    record "sauteeing onions" as achieved;
    say "You've got the onions off heat before they can scorch -- a good sign."

Instead of taking the onions when the onions are in the pan: try taking the pan.

Instead of smelling in the presence of the smoke:
    say "The scent of the late disaster lingers in the air."

Reconstituting the Apricots is a scene.

Reconstituting the Apricots begins when the dried apricots are in a container which contains boiling water.

When Reconstituting the Apricots begins:
    say "The apricots slowly begin to plump up again."

Reconstituting the Apricots ends when the dried apricots are not in a container which contains boiling water.

Every turn:
    if dried apricots have been in a container which contains boiling water for ten turns:
        say "The apricots have turned plump(ish).";
        now the apricots are reconstituted;
        record "reconstituting apricots" as achieved.

Test sautee with "peel onions / cut onions / get onions / put onions in pan / get sage / wash sage / cut sage / wait / wait / wait / wait / wait / wait / wait / get pan".

Test apricots with "get kettle / open kettle / get apricots / put apricots in kettle / put kettle on stove / wait".

Test chicken with "open refrigerator / get chicken / open drawer / get mallet / hit breast".

Test cheese with "get scissors / get cheese / unwrap cheese".

Test me with "full score / test sautee / full score / test

```
apricots / full score / test chicken / full score / test
cheese / full score".
```

And... at that point you're a lot less close to being done than you think. The filling -- onions, sage, apricot, and cheese -- must be assembled and put in the chicken breasts; these tied up in string; each roll dipped in egg yolk and rolled in panko crumbs; these arranged in the Calphalon pan and baked. Then later, the whole retrieved from the oven, and the breasts transferred to a plate while we deglaze the pan and concoct the sauce with the chicken broth, wine, butter, etc. Then the chicken is sliced and plated, and the sauce poured over top. Usually one also wants a side dish or two. A number of things can go interestingly wrong in this process, of course, and implementing it would require, among other things, an intelligent management of all the possible mixtures that result.

---

281 ExampleFarewell *RB*

People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.

We begin with the idea that each person comes with his own table of things to say:

```
"Farewell"


A person has a table name called conversation.


Instead of asking someone about something:
    let the source be the conversation of the noun;
    if topic understood is a topic listed in source:
        if there is a turn stamp entry:
            say "[The noun] has already told you that
[summary entry].";
        otherwise:
            now turn stamp entry is the turn count;
            say "[reply entry][paragraph break]";
    otherwise:
        say "[The noun] stares at you blankly."
```

For the sake of simplicity, we'll conflate asking and telling here, though it would certainly be possible to have a more complex implementation if we want the characters to be told things as well.

```
Instead of telling someone about something:
    try asking the noun about it.
```

Now we might want to add a recap command to review conversation that has already occurred.

```
Definition: a person is other if it is not the player.
```

```
Understand "recap" or "recall" or "review" as recalling
conversations.
```

```
Recalling conversations is an action applying to nothing.
```

Since we've been recording the turn count of each conversation bit, we can even present these in order by sorting the tables first.

```
Carry out recalling conversations:
    repeat with speaker running through other people:
        let source be the conversation of the speaker;
        sort source in turn stamp order;
        say "[The speaker] has so far told you: [line
break]";
        let index be 0;
        repeat through source:
            if there is a turn stamp entry:
                let index be 1;
                say " [summary entry][line break]";
        if index is 0, say " absolutely nothing[line break]";
        say conditional paragraph break.
```

Now it remains only to create a couple of characters and provide them both with something to say:

The Farewell Bend Cafe is a room. "Beautiful Farewell Bend, Idaho -- or is it Oregon? An almost-abandoned truckstop, in any case, on one of those interminable American east-west highways."

Tina is a woman in the Farewell Bend Cafe. The conversation of Tina is the Table of Tina's Chatter. "Tina the waitress is slowly pouring coffee from the pot with a black neck into the pot with an orange neck."

George is a man in the Farewell Bend Cafe. The conversation of George is the Table of George's Chatter. "There is also a large man at table five. The tattoo on his arm says George. For the moment we will assume that it is his own name and not someone else's."

Table of Tina's Chatter

[illegible]	[illegible]
[illegible]	"[illegible] is a vulture"
[illegible]	"[illegible] something doesn't have a tail.'"
[illegible]	"[illegible] explains gently, as to a child."

Table of George's Chatter

[illegible]	[illegible]
[illegible]	looking up."
[illegible]	[illegible]
[illegible]	[illegible]

A word of warning: this system does assume that every person in the game has a conversation table defined. If that were not the case, we would have to be a bit more careful.

As always, we can override specific words, too:

Instead of asking Tina about "advertisement" for the first time:
    say "Tina looks embarrassed. 'Of course! I almost forgot.' She hands you a brochure.";
    move the brochure to the player.

The encyclopedia sales brochure is a thing. The description is "A glossy flyer indicating that you can receive a free Volume A-Aalto of the New Idahoan Encyclopedia Set if you send back the business reply card, and then have the option of purchasing the remaining volumes at a very very reasonable price."

Test me with "recap / ask tina about aasvogel / recap / ask george about baccaceous / ask tina about absorptiometer / recap / ask tina about advertisement / read brochure".

---

282 ExampleSweeney

RB

A conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

"Sweeney"

A subject is a kind of thing. Some subjects are defined by the Table of Conversation Subjects.

Table of Conversation Subjects

subject	conversation
pets	Table of Pie Queries

Table of Job Queries

Understand "job" as employment. Understand "meat" or "food" as pies.

Table of Job Queries

"Say, are you hiring?" you ask, as casually as you can manage. [The interlocutor] looks you over dubiously. 'I might be hiring someone, but I can't say it would necessarily be you.'"	
"What happened to the last boy that worked here?" She shrugs. 'Young men these days are so unstable. He left-- who knows where he's gone? I haven't seen hair or fingernail of him for weeks.'"	

with 3 blank rows.

Table of Pie Queries

"What pies do you have available today, Mrs Lovett?' you ask. She starts, then smiles. 'Meat pies, of course.'"	

with 3 blank rows.

Table of Pie Flavor Queries

"What kind of meat goes into these pies, Mrs Lovett?' you ask pressingly. She looks shifty. 'Whatever the butcher brings this week,' she says. 'With the price of meat what it is, when you get it, you have to be glad of what you can get. If you get it.'"	

To copy (first table - a table name) to (second table - a table name):

```
    repeat through first table:
        let copied quip be "blank";
        if there is a quip entry, now the copied quip is the
quip entry;
        let copied discussion be "blank";
        if there is a discussion entry, now the copied
discussion is the discussion entry;
        let copied subtopics be second table;
        if there is a subtopics entry, now the copied
subtopics are the subtopics entry;
        choose a blank row in the second table;
        if copied quip is not "blank", now quip entry is
copied quip;
        if copied discussion is not "blank", now discussion
entry is copied discussion;
        if copied subtopics is not second table, now
subtopics entry is copied subtopics.



Current conversation table is a table name that varies.
Current conversation table is Table of Job Queries.



Interlocutor is a person that varies.



Understand "ask [someone] about [any subject]" as
asking it about the subject.



Asking it about the subject is an action applying to two
visible things.



Carry out asking it about the subject:
    say "You can't think of anything to say."



Instead of asking someone about the subject a subject
listed in the Table of Conversation Subjects:
    now interlocutor is noun;
    now current conversation table is the conversation of
```

```
the second noun;
    if the number of filled rows in the current conversation
table is 1:
        repeat through current conversation table:
            now label entry is 1;
        now number understood is 1;
        try selecting 1 instead;
    if the number of filled rows in the current conversation
table is 0:
        say "You can think of nothing further to say on that
topic.";
        stop the action;
    otherwise:
        let index be 0;
        let total be the number of filled rows in the current
conversation table;
        say "Do you mean ";
        repeat through current conversation table:
            now index is index + 1;
            now label entry is index;
            say "([index]) [quip entry]";
            if index is total, say "?";
            if index is total - 1, say ", or ";
            if index is less than total - 1, say ", ".



Understand "[number]" as selecting.



Selecting is an action applying to one number.



Carry out selecting:
    say "No such option is available."



Instead of selecting a label listed in the current
conversation table:
    say "[discussion entry][paragraph break]";
    if there is a subtopics entry:
        copy subtopics entry to current conversation table;
    choose row with label of number understood in the
current conversation table;
    blank out the whole row.
```

```
Mrs Lovett's Meat Pies is a room. Mrs Lovett is a
woman in Meat Pies.
```

```
Test me with "ask lovett about pies / ask lovett about
employment / 1 / 2 / ask lovett about pies".
```

---

283 **Example** **Introduction to Juggling** *RB*

Assortment of equipment defined with price and description, in a table.

Suppose we have a whole catalog-full of equipment that the player might want to purchase and use. We'll start by defining our purchasing rules:

```
"Introduction to Juggling"
```

We allow things to have prices, and the player's price to indicate how much money the player has:

```
Section 1 - Mail-ordering defined
```

```
Price is a kind of value. $100.99 specifies a price.
```

```
The player has a price. The price of the player is
$60.00.
```

Because we're allowing the player to order things that he can't currently see, we need to borrow a special kind of grammar from the Understanding chapter. All our orderable items in this example are toys, so "any toy" means any object of the toy kind, whether or not it is in view at the moment:

```
Understand "buy [any toy]" as ordering. Understand the
command "order" as something new. Understand the
command "order" as "buy".


Ordering is an action applying to one visible thing.


Check ordering:
    if the cost of the noun is greater than the price of the
player, say "You only have [price of the player], while
[the noun] would cost [cost of the noun]." instead.


Carry out ordering:
    move the noun to the player;
    decrease the price of the player by the cost of the
noun.


Report ordering:
    say "You order [a noun], which is delivered instantly."
```

We should also handle the situation where the player orders another of something he has already bought and which is right in front of him:

```
Instead of buying something:
    say "You already have [a noun]."
```

So much for the general rules for this scenario. Now we move on to particulars: the actual items the player is allowed to order. Each item will have a description, a price, and a difficulty representing how skilled the player must be in order to make use of that item.

Since we are going to use price and difficulty in the table that defines our juggling equipment, we need to mention these kinds of value before the line that says how toys are defined.

Section 2 - The Scenario

Difficulty is a kind of value. The difficulties are easy, moderate, hard. The player has a difficulty. The difficulty of the player is easy.

The plural of toy is toys. A toy is a kind of thing. Some toys are defined by the Table of Juggling Equipment.

Table of Juggling Equipment

[illegible]

[illegible] moving color."

[illegible]

[illegible] of your hands."

[illegible]

backs of your hands, rolls up to your wrist, snaps through the air--all apparently of its own accord."

[illegible]

[illegible]

[illegible] with basic competence."

[illegible]

[illegible] to be visible from a stage."

[illegible]

[illegible] than necessary."

Notice that we are allowed to define "description" and other already-known properties in the table as well.

Backstage is a room. "A muffled black room with felt on the floors and walls. A glowing sign over the stage door says SHOW IN PROGRESS."

The Juggling Equipment Catalog is a thing in Backstage.

```
Instead of examining the Catalog:
    say "You read through the offerings, including:
[paragraph break]";
    repeat through Table of Juggling Equipment:
        say "[bold type][toy entry][roman type]: [description
entry] [cost entry], [restriction entry]. [paragraph break]".



When play begins:
    now right hand status line is "Budget: [price of the
player]";
    now left hand status line is "[location], feeling [if the
difficulty of player is easy]incompetent[end if][if the
difficulty of player is moderate]moderately skilled[end if]
[if the difficulty of player is hard]highly skilled[end if]".
```

And of course this will be no fun unless the player is allowed to use the equipment:

```
Understand "juggle [something]" as juggling.



Juggling is an action applying to one thing.



Check juggling:
    if the noun is not a toy listed in the Table of Juggling
Equipment, say "You can't juggle [a noun]!" instead;
    if the difficulty of the noun is greater than the difficulty
of the player, say "You're not quite ready to juggle
something like [the noun]. Better to start with an easier
toy." instead.



Carry out juggling:
    if a random chance of 1 in 3 succeeds:
        if the difficulty of the player is less than hard and
the difficulty of the player is the difficulty of the noun:
            now the difficulty of the player is the difficulty
after the difficulty of the player.
```

```
Report juggling:
    say "[outcome of the noun][paragraph break]".

Instead of burning the fireball set:
    say "It will flame by itself when you use it."

Test me with "read catalog / buy economy / buy
beanbag / juggle economy / juggle beanbag / juggle
beanbag / juggle beanbag / juggle beanbag / juggle
beanbag / juggle beanbag / juggle economy / juggle
economy / juggle economy / buy fireball set / juggle
fireball".
```

Example **Food Network Interactive** *RB*

Using a menu system from an extension, but adding our own material to it for this game.

"Basic Help Menu" is an extension that uses tables to provide a menu of instructions. Suppose we wanted to include this menu in our own game, but add a few custom menu items of our own:

```
"Food Network Interactive"

Include Basic Screen Effects by Emily Short. Include
Menus by Emily Short. Include Basic Help Menu by
Emily Short.

Table of Basic Help Options (continued)

[illegible]
[illegible]
[illegible]
"If you have any difficulties with [story title], please contact me at
fakeaddress@gmail.com."
```

This table is one that is pre-defined by the extension. By continuing it, we add a few additional items to the list.

And since we've promised a sub-menu of recipes:

```
Table of Recipes

[illegible]
"Salmon Tartare" buy extremely fresh salmon. Raw fish should be
served on the day it was caught, if possible. To guarantee this,
visit an Asian supermarket or specialty store, and buy salmon
marked 'sashimi grade'..."
"Pecan Brownies" half a pound of pecans..."



Whole Foods is a room.
```

To test it, type HELP and then experiment.

---

285 **ExampleTrieste** *RB*

Table amendment to adjust HELP commands provided for the player.

Suppose we are using an extension in which another author has defined some help topics for the player, and we want to amend them for our game.

We'll start with the portion of the text that we have inherited from the extension:

```
"Trieste"



Section 1 - Procedure
```

A help-topic is a kind of value. Some help-topics are defined by the Table of Standard Instructions.

Table of Standard Instructions

help-topic	reply
"Commands"	"This game recognizes 150 common commands for forms of military attack. These include..."
"Saving"	"To save the game, type SAVE. You will be prompted to supply a file-name for your saved game. If you'd like to return to play at that point again later, RESTORE the saved game."

Understand "help [help-topic]" as asking for help about.
Asking for help about is an action out of world, applying to one help-topic.
Understand "help" or "help [text]" as a mistake ("Help is available on the following topics: [help-topics list]").

To say help-topics list:
    repeat through the Table of Standard Instructions:
        say "[line break] [help-topic entry]";

Carry out asking for help about:
    repeat through the Table of Standard Instructions:
        if the help-topic understood is the help-topic entry:
            say "[reply entry][paragraph break]";
            break.

Section 2 - Scenario

Now, let's imagine our game is a special one in which only a very limited supply of moves are allowed. In that case, we'll want to replace the information on commands:

Table of Standard Instructions (amended)

help topic

"The only commands this game recognizes are HOLD, MOVE, CONVOY, SUPPORT MOVE, and SUPPORT HOLD. No others are necessary."

Board Room is a room. Mark is a man in the Board Room. "Russia (played by Mark) is also hovering over the board."

Guest Bathroom is south of Board Room. Lena and Rob are in the Guest Bathroom. Lena is a woman. Rob is a man.

Rule for writing a paragraph about Lena when Lena is in the Guest Bathroom and Rob is in the Guest Bathroom:
    say "[Lena] (Italy) and [Rob] (Great Britain) are having a hushed conversation while leaning against your good towels. They stop and stare at you when you come in."

Test me with "help / help commands / help saving".

Chapter 17: Understanding

ls consisting only of nouns; *§17.11. Understanding values*; *§17.12. This/that*; *§17.13. New tokens*; *§17.*

Contents of *Writing with Inform*
Chapter 16: Tables
Chapter 18: Activities
Indexes of the examples

§17.1. Understand

During play, the computer and the player alternate in writing messages to each other: in the player's case, these are short instructions, usually saying what to do next. A wide range of such "commands" are automatically understood, but these only apply to the standard built-in actions. (This wide range is conveniently tabulated in the Commands part of the Actions index.) If we want the player to be able to command new actions, then we need to specify what is to be understood as what. For this, we supply special sentences starting with the word "Understand".

Suppose we return to the earlier example of a newly created action:

```
Photographing is an action applying to one visible thing and requiring
light.
```

We then supply lines of grammar (as they are called) for Inform to recognise, like so:

```
Understand "photograph [someone]" as photographing.
```

```
Understand "photograph [an open door]" as photographing.
```

As usual, the square brackets indicate something which stands for text, rather than text to be taken verbatim. "[someone]" needs to be the name of anything of the kind "person", for instance (though as usual that person will need to be in sight of the player for the name to be accepted). The first word - in these examples "photograph" - must be something definite, not a substitution like this.

For obvious reasons, this pattern of words needs to match the expectations of the action. Photographing applies to "one visible thing" - the "visible" just means it does not need to be touched, only seen - so neither of these would be allowable:

```
Understand "photograph" as photographing.
Understand "photograph [someone] standing next to [something]" as
photographing.
```

The first is probably bad because it supplies no things at all, the second is certainly because it supplies two: what we want, of course, is just the one. (The reason the first is only probably bad is that it's possible to tell Inform how to choose the object if the player doesn't: see the "supplying a missing noun" activity.)

Start of Chapter 17: Understanding

Back to Chapter 16: Tables: §16.19. Table amendments

Onward to §17.2. New commands for old grammar

Example 286: ★ **Indirection** Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

Example 287: ★ **XYZZY** Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

Example 288: ★★ **Xylan** Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

---

§17.2. New commands for old grammar

In the photography example, we are providing entirely new grammar for an action not ordinarily built in to Inform. But we often want simply to provide alternative grammar for existing actions, or even to put new interpretations on commands that Inform already recognises. For instance:

```
Understand "deposit [something] in [an open container]" as inserting
it into.
```

The inserting action is built in to Inform, but the command "deposit" is not, so this is created as new. It is occasionally useful to put a twist on this:

```
Understand "fill [an open container] with [something]" as inserting it
into (with nouns reversed).
```

The clause "(with nouns reversed)" tells Inform to exchange the two nouns parsed, which is necessary because the inserting action expects the noun to be the item and the second noun to be the container, not vice versa.

The following example:

```
Understand "access [something]" as opening.
```

might look as if it makes "access" behave just like "open" when the player types it, but that's not so: "open" can also be used in constructions like "open the door with the brass key", in which case it is understood as the unlocking action. We could add another line to make "access" behave this way too, but if what we really want is to make "access" behave just like "open", it's easier simply to say so:

```
Understand the command "access" as "open".
```

This is very useful when adding a new command which needs synonyms:

```
Understand the commands "snap" and "picture" as "photograph".
```

We can check the current stock of commands by looking at the table in the Actions index: for instance, before making "snap" synonymous with "photograph", it might be wise to check that it is not already defined as a command for breaking something.

Start of Chapter 17: Understanding
Back to §17.1. Understand
Onward to §17.3. Overriding existing commands
Example 289: ★ **Alpaca Farm** A generic USE action which behaves sensibly with a range of different objects.
Example 290: ★ **Anchorite** By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.
Example 291: ★★★★ **Cloak of Darkness** Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

---

§17.3. Overriding existing commands

Suppose we are devising specialist commands for a game of whist, and we want "discard" as one of them. Looking at the table of commands in the Action index, we find that, inconveniently enough, "discard" already has a meaning: it is synonymous with "drop", and while that might be sensible most of the time, it is perfectly wrong now. We need a way to free up "discard" for our own use. We can do that by:

```
Understand the command "discard" as something new.
```

This cuts it loose, so to speak, and ready to be given new meanings. If we check the Actions index again, we find no mention of "discard" - it is now a blank slate - but "drop" is still exactly as it was. We could now say something like:

```
Understand "discard [something]" as discarding.
```

(If we had declared that "drop" was something new, the whole thing would have happened in reverse, with "discard" retaining all of the original grammar. Inform does not distinguish between a command and its synonym.)

The "... as something new" sentence works even for a command which did not exist anyway, for instance with:

```
Understand the command "zylqix" as something new.
```

Of course this does nothing - but it is intentional that it generates no problem messages: it means that the sentence can be used to force a command to be fresh and untouched by previous definitions, which might be useful when working with extensions by other people.

It is also possible to clear out all the commands leading to a given action:

```
Understand nothing as taking.
```

The commands "take" and "get" will still exist, but now they'll only have their other senses (for taking off clothes, for getting out of boxes).

Start of Chapter 17: Understanding
Back to §17.2. New commands for old grammar
Onward to §17.4. Standard tokens of grammar

Example 292: ★ **The Trouble with Printing** Making a READ command, distinct from EXAMINE, for legible objects.

Example 293: ★★ **Lanista 2** Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an

ATTACK IT WITH action is created to replace regular attacking. Also folds a new DIAGNOSE command into the system.

---

§17.4. Standard tokens of grammar

We have already seen "[something]" and "[someone]", which are standard examples of "tokens of grammar" - patterns matched by suitable named things. There are several other standard tokens, provided not so much from necessity but to allow the story parser to be more graceful and responsive. "[someone]" matches the same possibilities as "[a person]" would, but the parser handles it a little better in cases of failure. These special tokens are best explained by looking at some of the examples in the standard grammar, which can be browsed in the Index of any story.

```
Understand "wear [something preferably held]" as wearing.
```

Here we expect that the named item will be one that is held by the player, and the parser will use this to resolve ambiguities between names of things carried and not carried. (If the action is one which positively requires that its noun be something carried, a command matching this token against something not carried will generate an automatic attempt to take it.)

```
Understand "take [things]" as taking.
Understand "drop [things preferably held]" as dropping.
```

"[things]" is like "[something]" but allows a list of items, or a vague plural like "all", to be typed. The result will be a sequence of actions, one for each item thus described. "[things preferably held]" is the analogous token for "[something preferably held]".

```
Understand "take [things inside] from [something]" as removing.
```

"[things inside]" matches only what is inside the second-named thing, and ensures that (for instance) the command "take all from box" does not also try to take the box.

```
Understand "put [other things] in/inside/into [something]" as inserting
```

it into.

Similarly, "[other things]" will allow anything except the second-named thing. (Like "[things inside]" it is really only needed for handling containers.)

Finally there is "[any things]", which should be used only with care. This is like "[things]" but with no restriction at all on where the item comes from: it might be invisible, or from a different room, or out of play altogether. If we use this, we had better remember that it would match ALL, with quite extravagant consequences.

Start of Chapter 17: Understanding
Back to §17.3. Overriding existing commands
Onward to §17.5. The text token

Example 294: ★ **Shawn's Bad Day** Allowing the player to EXAMINE ALL.

Example 295: ★★★ **The Left Hand of Autumn** The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple examination command that describes groups in special ways.

---

§17.5. The text token

Most actions involve items: taking a vase, perhaps. As we shall see, they might also involve values, or a mixture of the two: turning a dial to 17 would involve both a thing (the dial) and a number (17). A few of Inform's built-in actions, however, can act on any text at all. For instance, asking the Sybil about the Persian army would involve a thing (the Sybil) and some text ("Persian army"). Inform does not try to understand automatically what that text might mean, or to relate it to any items, places or values it knows about: instead, Inform leaves that to the specific story to work out for itself, since the answer is bound to depend on the context. (In the chapter on Tables, we saw ways to compile tables of responses to particular topics of conversation.)

The token for "accept any text here" is just "[text]". For instance, if we create an action with:

```
Getting help about is an action applying to one topic.
```

We can then provide grammar for this action like so:

```
Understand "help on [text]" as getting help about.
```

When text like this is successfully matched, it is placed in a value called "the topic understood". (The term "topic" is used traditionally, really: most of the times one needs this feature, it's for a topic of conversation, or a topic being looked up in a book.)

The fact that "[text]" can match anything means that it's difficult to tell which version of a command was intended if they disagree only from a "[text]" onwards. For example, given:

```
Yelling specifically is an action applying to one topic. Understand "yell
[text]" as yelling specifically. Understand "yell [text] at/to [someone]"
as answering it that (with nouns reversed).
```

...Inform will in fact try the second possibility first, as being the more specific, but the result may freeze out the first possibility altogether due to autocompletion of commands.

Start of Chapter 17: Understanding

Back to §17.4. Standard tokens of grammar

Onward to §17.6. Actions applying to kinds of value

Example 296: ★ **Ish.** A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

Example 297: ★★ **Nameless** ASKing someone about an object

rather than about a topic.

§17.6. Actions applying to kinds of value

Almost all actions apply to things: the player picks them up, pushes them, looks at them and so on. We only occasionally need to recognise other kinds of value, but when we do, we can. For instance:

```
Adjusting it to is an action applying to one thing and one number.

Understand "adjust [something] to [a number]" as adjusting it to.
```

The substitution "[a number]" matches any number (actually any whole number that is not too large) typed by the player. Inform checks the various kinds being used to make sure that everything matches, so, for instance, this would be disallowed:

```
Understand "adjust [something] to [something]" as adjusting it to.
```

Start of Chapter 17: Understanding

Back to §17.5. The text token

Onward to §17.7. Understanding any, understanding rooms

Example 298: **Safety** A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.

Example 299: **Tom's Midnight Garden** A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running.

Time on the face is also reported differently depending on whether the clock is analog or digital.

Example 300: ★★ **Ibid.** A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

---

§17.7. Understanding any, understanding rooms

Ordinarily, if we write

```
Understand "manipulate [something]".
```

then the "[something]" will only match what is within reach or sight: this is the concept of "scope", which is what prevents a player from spookily acting on objects from a distance. The parser itself prevents the manipulation rules from ever being invoked on such distant items, which is as it should be.

Sometimes, though, we positively want to allow this possibility. If we use the special word "any", as in

```
Understand "manipulate [any door]".
```

then any door, anywhere in the model world, can be allowed in the player's command. (Of course, the manipulation rules may not do what the player hopes: all that has happened is that the command is now possible to type.) The "any" can be followed by any description of items or rooms, and the latter opens up new possibilities, since rooms are ordinarily never allowed to be named in the player's commands.

For example, the following gives the player the ability to walk between rooms without giving explicit directions of movement.

```
Going by name is an action applying to one thing.

Carry out going by name: say "You walk to [the noun]."; move the
player to the noun.

Understand "go to [any adjacent visited room]" as going by name.
```

(This is really only a sketch: in a finished work, "go to" would produce helpful errors if non-adjacent but visited rooms were named, and we might also worry about rules applying to movement, because the method above will circumvent them.)

As might be expected, "[anything]" means the same as "[any thing]"; "[anybody]" and "[anyone]" mean the same as "[any person]"; and "[anywhere]" means the same as "[any room]".

Start of Chapter 17: Understanding

Back to §17.6. Actions applying to kinds of value

Onward to §17.8. Understanding names

Example 301: ★ **One of Those Mornings** A FIND command that allows the player to find a lost object anywhere

Example 302: ★★ **Actaeon** A FOLLOW command allowing the player to pursue a person who has just left the room.

§17.8. Understanding names

So far in this chapter, Understand sentences have been used to give names to actions, but they can also be used to name objects - in particular, things and rooms.

This normally happens automatically. For instance, writing

```
The St Bernard is an animal in the Monastery Cages.
```

makes ST BERNARD refer to the dog, and MONASTERY CAGES refer to the room. But sometimes, as here, that isn't really enough. Why shouldn't the player type EXAMINE DOG? One way to allow this is to write:

```
Understand "dog" as the St Bernard.
```

Matters become more complicated when the player wants to refer to more than one object at once. When a kind is created, and the source text constructs multiple duplicate items of that kind, Inform generates a plural of the kind's name in order to understand commands referring to these multiples. For instance, given...

```
The Lake is a room. A duck is a kind of animal. Four ducks are in
    the Lake.
```

...the player can type TAKE DUCKS to try to pick up all four.

Once again the automatic behaviour can be enhanced:

```
Understand "birds" and "ruddy ducks" as the plural of duck.
```

Now TAKE BIRDS and TAKE DUCKS are equivalent. Plurals can even, strange as it may seem, be given for single things:

```
The magpie is in the Lake. Understand "birds" as the plural of the
magpie.
```

And now TAKE BIRDS tries to take all four ducks and the magpie too.

Start of Chapter 17: Understanding
Back to §17.7. Understanding any, understanding rooms
Onward to §17.9. Understanding kinds of value

§17.9. Understanding kinds of value

In many cases, if K is the name of a kind of value, then Inform automatically makes an Understand token called "[K]" which matches only values of K. An example is "[number]", which matches text like 203 or SEVEN. There is a chart of the kinds of value in the Kinds index for a project, showing which ones can be understood in this way.

In particular, any newly created kind of value can always be understood. We make good use of that in the example story "Studious":

```
"Studious"

The Studio is a room. "The unreal world of the photographic studio, full of fake furniture, cantilevered stands and silver-white shades like miniature parachutes." The lumpy black camera is in the Studio. "A lumpy black camera hangs from a tripod."

The rake-thin model is a woman in the Studio. "A rake-thin model, exquisitely bored and boringly exquisite, angles herself indolently."

Limb is a kind of value. The limbs are left leg, left arm, right leg and right arm.

Detailing is an action applying to one limb and one visible thing, requiring light. Check detailing: if the camera is not carried then say "You can hardly photograph without a camera, now can you?" instead. Report detailing: say "Click! You take a detail photograph of the [limb understood] of [the second noun]."

Understand "photograph [limb] of [a person]" as detailing.
```

```
Test me with "get camera / photograph left leg of model".
```

Note the way we can refer to the limb mentioned by the player as the "limb understood". Similarly, we could talk about the "number understood" if the value parsed had been a number, and so on.

One of the built-in kinds of value is worth special note: time. A time can hold either a specific time of day, such as 10:23 PM, or a duration of something, such as 21 minutes. The "[a time]" token matches times of day, such as 10:15 AM or MIDNIGHT. But 10 MINUTES wouldn't be recognised by "[a time]" since it isn't a specific moment in the day. To get around this, an alternative version called "[a time period]" is available. So:

```
Understand "wait for [a time period]" as ...
```

would match WAIT FOR AN HOUR or WAIT FOR TWO HOURS 12 MINUTES.

Start of Chapter 17: Understanding

Back to §17.8. Understanding names

Onward to §17.10. Commands consisting only of nouns

Example 303: ★ **Pages** A book with pages that can be read by number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

Example 304: ★★ **Down in Oodville** Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

Example 305: ★★★ **Straw Into Gold** Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or

"man" -- depending on which the player last used -- until the first time the player refers to him as "Rumpelstiltskin".

§17.10. Commands consisting only of nouns

In every example so far, and in almost all practical cases, the first word in a command which results in an action will be something fixed: a verb, in fact. When we write

```
Understand "photograph [something]" as photographing.
```

we are saying that the first word of such a command will always be "photograph". Occasionally, though, we would like to understand a noun as a command, perhaps in a situation where the command is obvious. If we say:

```
Understand "[something]" as examining.
```

then the command "examine" will be implicit when the player types a bare noun:

```
A red box and a blue ball are here.
> BALL
The blue ball is plaited from many small leather patches.
```

so that the command "ball" has resulted in the action "examining the blue ball".

This is a feature which should be used sparingly, since it could easily lead to confusion if not carefully explained to the player. By default, it is not used at all.

It also has what may be a serious limitation: verbless commands like this work only when typed by the player as actions to follow - they do not work as instructions for other people. So for instance SVEN, BALL would not ask Sven to try examining the ball - instead it would generate the action "answering ball to Sven". (This is because the Inform parser decides whether PERSON, SOME TEXT is a request or just conversation by looking at the first word after the comma to see if it's a command.)

Start of Chapter 17: Understanding
Back to §17.9. Understanding

kinds of value

Onward to §17.11. Understanding values

Example 306: ★ **Misadventure** A going by name command which does respect movement rules, and accepts names of rooms as commands.

Example 307: ★★ **Safari Guide** The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling all openable doors on the way.

---

§17.11. Understanding values

"Understand" can be used to supply new ways to talk about both things and other values. For instance, if we create:

```
A brass lantern is in the Building.
```

then it can be called "brass", or "lantern", but not "lamp": Inform does not really know what these words mean, and has no grasp of synonyms. We can arrange for "lamp" to work as well like so:

```
Understand "lamp" as the lantern.
Understand "old lamp" as the lantern.
```

With care, we can do the same trick for entire kinds of thing at once. It is not ordinarily the case that a thing can be called by the name of its kind: if we put a woman called April into a room, then she can usually be called "April", but not "woman". (The exception is when we do not specify any name for her - in that case, Inform will give up and call her just "woman".) So there is not usually any form of words which can refer to anything of a given kind. If we should want this, we have to say so explicitly:

```
Understand "machine" as a device.
```

Device is a kind, so now the word "machine" can be used to refer to any device: if there are two in the same place, the result might play out like so:

```
>switch machine on
Which do you mean, the bale twiner or the grain thresher?
>twiner
You watch absorbed as a perfect cube of hay is trussed up like a
parcel.
```

Similarly, we might conceivably want to allow new ways to recognise values - in this case, a number:

```
Understand "eleventy-one" as 111.
```

When making complicated names, we need to watch out for the possibility of writing a definition which will cause Inform to go around in circles (something which will show up as a "Too many activities at once" run-time problem). For instance,

```
Understand "[thing] substitute" as the placebo.
```

will fail because Inform, working left to right, needs to look for every possible object name before it can progress: one possibility is the placebo itself: to check that, it needs to look for every possible object name: and so on, never finishing. A definition like this one very likely matches too much in any case (would we really want to accept PLACEBO SUBSTITUTE or CIGARETTE SUBSTITUTE SUBSTITUTE SUBSTITUTE here, as the definition implies?).

Start of Chapter 17: Understanding
Back to §17.10. Commands consisting only of nouns
Onward to §17.12. This/that

Example 308: ★ **Palette** An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean")

are accepted alongside everyday ones ("blue").

Example 309: ★★★ **Baritone, Bass** Letting the player pick a gender (or perhaps other characteristics) before starting play.

---

§17.12. This/that

We have already seen "or" used in "Understand" sentences:

```
Understand "scarlet" or "crimson" as red.
```

In general, any number of alternative forms can be given which are to be understood as the same thing (in this case the colour red). When the alternatives are in any way complicated, "or" should always be used, but a shorthand form is allowed for simple cases where it is only a matter of a single word having several possibilities:

```
Understand "reach underneath/under/beneath [something]" as
looking under.
```

This is shorthand for:

```
Understand "reach underneath [something]" or "reach under
[something]" or "reach beneath [something]" as looking under.
```

Which in turn is shorthand for:

```
Understand "reach underneath [something]" as looking under.
Understand "reach under [something]" as looking under. Understand
"reach beneath [something]" as looking under.
```

It's possible also to make that second word optional:

```
Understand "reach underneath/under/beneath/-- [something]" as
looking under.
```

because "--" is read by Inform as "no word at all". If "--" is an option, it can only be given once and at the end of the list of possibilities.

To recapitulate: the slash "/" can only be used between single, literal words, and is best for the wayward prepositions of English ("in/into/inside", and so forth). For anything more complex, always use "or".

Start of Chapter 17: Understanding
Back to §17.11. Understanding values
Onward to §17.13. New tokens

§17.13. New tokens

We have now made good use of square-bracketed tokens, such as "[something]", in a variety of "Understand..." sentences. It is sometimes convenient to create new tokens of our own, to match whatever grammar we choose: this enables complicated knots of grammar to be used in many different "Understand..." sentences without having to write it all out each time.

For instance, here are new tokens: one for each of two groups of alternative prepositions.

```
Understand "beneath/under/by/near/beside/alongside/against" or
"next to" or "in front of" as "[beside]".


Understand "on/in/inside" or "on top of" as "[within]".
```

Again, note that the slash indicates a choice between words only, not between entire phrases. For instance, if we write:

```
Understand "red bird/robin" as "[robin]".
```

then the two alternative forms are "red bird" and "red robin", not "red bird" and "robin". By contrast,

```
Understand "red bird" or "robin" as "[robin]".
```

will understand either "red bird" or "robin" but not "red robin". If we want to capture all three forms, we might define

```
Understand "red bird/robin" or "robin" as "[robin]".
```

Start of Chapter 17: Understanding
Back to §17.12. This/that
Onward to §17.14. Tokens can produce values
Example 310: **Lies**
Commands to allow the player to lie down in three different ways.

§17.14. Tokens can produce values

The examples just seen were tokens which simply matched specific words typed by the player, but newly created tokens can also produce values:

```
Colour is a kind of value. The colours are red, green and blue.
Understand "colour [a colour]" or "[a colour] shade" as "[tint]".
```

Here the "[tint]" token matches, for instance, "colour red" and "blue shade", which would result in the values red and blue, respectively.

Tokens are not allowed to produce more than one value, and if several patterns are given to define them then those patterns have to be compatible. That means the following is disallowed, since it might work out to a colour, or to an object, leaving Inform unable to judge whether an action can safely be applied to the result.

```
Understand "colour [a colour]" or "[something]" as "[tint]".
```

§17.15. Understanding things by their properties

Items are ordinarily understood only by their original given names. For instance, if we have:

```
In the Herb Garden is a china pot.
```

then the player could refer to this as "pot", "china pot" or "china". We can embellish this by adding extra forms:

```
Understand "chinese pot" or "chinese vase" as the china pot.
```

But suppose the pot changes its nature in the course of play? If we have:

```
The china pot can be unbroken or broken. The china pot is unbroken.


After dropping the china pot:
    say "Crack!";
    now the china pot is broken;
    now the printed name of the pot is "broken pot".
```

So now the player would reasonably expect to call it "broken pot", a wording which would have been rejected before. We can achieve this by writing:

```
Understand the unbroken property as describing the pot.
```

which allows "unbroken" or "broken" to describe the pot, depending on its state. And, since the player might well use a different adjective but with the same idea in mind, we can even add:

```
Understand "shattered" or "cracked" or "smashed" as broken.
Understand "pristine" as unbroken.
```

This is something of a toy example, but the feature looks rather more useful when there are more pots than just one:

```
"Terracotta"


A flowerpot is a kind of thing. A flowerpot can be unbroken or broken.
Understand the broken property as describing a flowerpot.


After dropping an unbroken flowerpot:
    say "Crack!";
    now the noun is broken;
    now the printed name of the noun is "broken flowerpot";
    now the printed plural name of the noun is "broken flowerpots".


The Herb Garden is a room. In the Herb Garden are ten unbroken
flowerpots.
```

We then have the dialogue:

```
Herb Garden
You can see ten flowerpots here.
>get two flowerpots
flowerpot: Taken.
flowerpot: Taken.
>drop all
flowerpot: Crack!
flowerpot: Crack!
>look
Herb Garden
You can see two broken flowerpots and eight flowerpots here.
>get an unbroken flowerpot
Taken.
```

and so on and so forth.

There are in fact two slightly different forms of this kind of sentence:

Understand the broken property as describing a flowerpot.
Understand the broken property as referring to a flowerpot.

The only difference is that in the "describing" case, the property's name alone can mean the thing in question - so "take unbroken" will work; whereas, in the "referring to", the property's name can only be used as an adjective preceding the name of thing itself - so "take unbroken flowerpot" will work but "take unbroken" will not.

Start of Chapter 17: Understanding

Back to §17.14. Tokens can produce values

Onward to §17.16. Understanding things by their relations

Example 311: ★ **Aspect** Understanding aspect ratios (a unit) in the names of televisions.

Example 312: ★ **Hymenaeus** Understanding "flaming torch" and "extinguished torch" to refer to torches when lit and unlit.

Example 313: ★★ **Channel 1** Understanding channels (a number) in the names of televisions.

Example 314: ★★ **Terracottissima** The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

Example 315: ★★ **Peers** The peers of the English realm come in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince

- and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

Example 316: ★★★ **Channel 2** Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

Example 317: ★★★ **Terracottissima Maxima** Flowerpots with textual names that might change during play.

Example 318: ★★★ **Tilt 1** A deck of cards with fully implemented individual cards, which can be separately drawn and discarded, and referred to by name.

---

§17.16. Understanding things by their relations

Sometimes it makes sense for the name of something to involve the names of other things to which it is related. For instance, if we say TAKE THE BOTTLE OF WINE, we mean that the bottle currently contains wine - if it were the very same bottle containing water, we would call it something else.

For names which must involve related names, a special form of token is provided. For instance, we could say:

```
A box is a kind of container. Understand "box of [something related
by containment]" as a box.
```

```
The Toyshop is a room. The red box is a box in the Toyshop. Some
crayons are in the red box.
```

and now TAKE BOX OF CRAYONS will work, because CRAYONS matches against "[something related by containment]" for the red box - or it does for as long as the crayons are there. We can have similar matches against relations of all kinds, but have to name the relation explicitly. (See the examples at the end of this section for plenty of cases.)

We can also reverse the sense. If we write:

```
A box is a kind of container. Understand "box in [something related
by reversed containment]" as a box.
```

```
The Toyshop is a room. The crate and the hammock are in the
Toyshop. In the crate is a box. In the hammock is a box.
```

then TAKE THE BOX IN THE HAMMOCK will work: here, the relation goes the other way, because the box is being contained by the other-named item, rather than doing the containing.

Start of Chapter 17: Understanding

Back to §17.15. Understanding things by their properties

Onward to §17.17. Context: understanding when

Example 319: **Cinco** A taco shell that can be referred to (when it contains things) in terms of its contents.

Example 320: **Puncak Jaya** When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

Example 321: **Whither?** A door whose description says where it leads; and which

automatically understands references such as "the west door" and "the east door" depending on which direction it leads from the location.

Example 322: **Claims Adjustment** An instant camera that spits out photographs of anything the player chooses to take a picture of.

---

§17.17. Context: understanding when

We have now seen several different forms of "Understand" sentence: for instance,

```
Understand the colour property as describing a building block.
Understand "mix [colour] paint" as mixing paint.
Understand "rouge" as red.
Understand "curious girl" as Alice.
```

Any of these may optionally have a condition tacked on: for instance,

```
Understand "mix [colour] paint" as mixing paint when the location is
the Workshop.
Understand "rouge" as red when the make-up set is visible.
```

In principle, "when ..." can take in any condition at all. In practice a little care should be exercised not to do anything too slow, or which might have side-effects. (For instance, referring the decision to a phrase which then printed text up would be a bad idea.) Moreover, we must remember that the "noun" and "second noun" are not known yet, nor do we know what the action will be. So we cannot safely say "when the noun is the fir cone", for instance, or refer to things like "the number understood". (We aren't done understanding yet.) If we want more sophisticated handling of such cases, we need to write checking rules and so on in the usual way.

Contexts can be useful to make sense of things having different names depending on who is being spoken to, as here:

```
Understand "your" as a thing when the item described is held by the
person asked.
```

With this rule in place FRODO, GIVE ME YOUR RING means that Frodo will know which ring is meant, even if there are a couple of dozen other rings present.

If the name of something has to change completely, perhaps because the player's understanding of events has changed completely, then Inform's standard way of handling names can be a nuisance. When an item or room is created, Inform automatically makes its name understood as referring to it (in fact, it makes each individual word in that name understood). For instance,

```
The Wabe is a room. The blue peacock and the sundial are in the
Wabe.
```

means that the player can type EXAMINE BLUE PEACOCK or PUSH SUNDIAL or SHOWME WABE or TAKE BLUE, and so on. This is almost always a good thing, and here there's no problem, because peacocks and sundials are not usually disguised. But here is a case where a disguise is needed:

```
The secret document is a privately-named thing in the drawer.
The printed name of the secret document is "[if the secret document
is handled]secret document[otherwise]dusty paper".
Understand "dusty" and "paper" as the secret document.
Understand "secret" and "document" as the secret document when
the secret document is handled.
After taking the secret document for the first time: say "Heavens! It is
the secret document!"
```

As this demonstrates, the either/or property "privately-named" makes Inform create a thing or room which starts out with no automatic understandings at all. The name it happens to have in the source text is ignored. If we simply write:

```
The ungraspable concept is a privately-named thing in the Dining
Room.
```

then nothing the player can type will ever refer to it; though he will see it, and even be able to pick it up by typing TAKE ALL.

The reverse property is "publicly-named", which all things and rooms are by default.

Inform has four built-in kinds of object (room, thing, direction and region), and all of those have this either/or property. When we create new kinds, they're normally kinds

of those four fundamental ones, so they pick up the same behaviour. But if we create a new kind of object outside of these four, that won't be true unless we make it so:

```
A concept is a kind of object. A concept can be privately-named or
publicly-named. A concept is usually publicly-named.
```

(Privately-named is a property which only affects how Inform creates the object, and it can't usefully be given or taken away during play. "Understand ... when ..." is the way to change names during play.)

Start of Chapter 17: Understanding
Back to §17.16. Understanding things by their relations
Onward to §17.18. Changing the meaning of pronouns

Example 323: ★ **Quiz Show** In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

Example 324: ★★ **Bibliophilia** A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

---

§17.18. Changing the meaning of pronouns

The pronouns IT, HIM, HER and THEM are constantly adjusted during play, to save the player time when typing commands. If the player types EXAMINE NECKLACE on one turn, it's sufficient to type TAKE IT on the next, and IT will be understood as

meaning whatever NECKLACE meant last turn.

All of that happens automatically, but once in a while the result can be unfortunate. Suppose that when the player examines the necklace, a security system automatically drugs her unconscious, and she wakes up in a cell, hours later, and is told that the cell is bare except for a key on the floor. If she types TAKE IT, she clearly doesn't mean IT to mean the necklace any more; she means the key. Inform's parser can't make guesses like this, so the following phrase can be used to help it.

set pronouns from (object)

This phrase adjusts the meaning of pronouns like IT, HIM, HER and THEM in the command parser as if the object mentioned has become the subject of conversation. Example: the combination of

```
set pronouns from the key;
set pronouns from Bunny;
```

might change IT to mean the silver key and HIM to mean Harry "Bunny" Manders, while leaving HER and THEM unaltered.

Start of Chapter 17: Understanding
Back to §17.17. Context: understanding when
Onward to §17.19. Does the player mean...

Example 325: **Pot of Petunias** Responding sensibly to a pot of petunias falling from the sky.

§17.19. Does the player mean...

When the player types an ambiguous reference, we need to work out what is meant. Consider the following source text:

```
The Champs du Mars is a room. The great Eiffel Tower is here. "The
great Tower stands high over you." The souvenir model Eiffel Tower
is here. "Comparatively tiny is the souvenir version."
```

Now suppose the player types GET TOWER. The response will be:

```
Which do you mean, the great Eiffel Tower or the souvenir model
Eiffel Tower?
```

Which is a silly question, exposing our work of IF as something artificial. It's obvious to the author of the source text, and to the player, that the souvenir must be what is meant: but this is not obvious to the computer program running the story. Works of IF gain a subtle feeling of quality from being able to understand ambiguous references of the kind above, and Inform provides us with a way to do this by giving the parser clues in the form of "Does the player mean..." rules. For instance, if we add:

```
Does the player mean taking the great Eiffel Tower: it is very unlikely.
```

then the response to GET TOWER will now be:

```
(the souvenir model Eiffel Tower)
Taken.
```

"Does the player mean..." rules look at the actions which are possible interpretations of what the player typed, and grade them according to how likely they seem. (Note that these rules are only ever used to handle ambiguities: if the player unambiguously types GET GREAT EIFFEL TOWER, that will be the action. And the rules are only used where they are able to make a decision: if there are still multiple equally plausible meanings, the parser will ask about all possibilities, not just the most likely ones.) Rules in this rulebook can either decide nothing, or come up with one of the following verdicts:

```
it is very likely
it is likely
it is possible
it is unlikely
it is very unlikely
```

If there are no "does the player mean" rules, or the rules make no decision on a given possible action, it will be ranked as "it is possible".

We may use these rules to affect all sorts of interaction with a specific object or kind of object, as in

```
Does the player mean doing something with the cursed dagger of
Thog: it is very unlikely.
Does the player mean doing something with the cursed dagger of
Thog when the player is hypnotized: it is likely.
```

...and so on.

Notice that we can also make rules about actions that apply to two objects, so for instance:

```
Does the player mean throwing the can of shoe polish at the shoe
polish vending machine: it is likely.
```

which nicely clarifies THROW POLISH AT POLISH, but does not comment on the likelihood of throwing the can at other things or of throwing other things at the vending machine. Moreover, the (suspected) identity of the first item will be known when the rule is consulted; thus

```
Does the player mean tying the noun to the noun: it is very unlikely.
```

will tell Inform to prefer not to tie something to itself if other interpretations are available.

But there is a caveat. There are some cases where this mechanism will not in fact help Inform to choose its way out of an ambiguous command, because of the way it parses one noun at a time. It usually needs to understand the first noun before it will even try to make sense of the second. So a rule like:

```
Does the player mean throwing the can of shoe polish at the tree: it is
likely.
```

may not work if the player types THROW POLISH AT TREE and POLISH is ambiguous, because when the parser is trying to understand POLISH, it hasn't yet seen to the end of the command and realised that the second noun will be the tree; so the second noun is unset and the rule won't match.

As a caveat to the caveat, the "inserting it into", "removing it from" and "putting it on" actions have this slightly back to front. These are parsed using the (little-used) "[other

things]" or "[things inside]" tokens, and the Inform parser tries to detect the second noun before the first one, since the identity of the first has to depend on the second. So for instance if the situation contains "an oak tree" and also "an oak chest", we could write:

```
Does the player mean inserting into the oak chest:
    it is very likely.
```

which would successfully make PUT COIN IN OAK mean the chest, not the tree. (Note the way we write "inserting into" without saying anything about what's being inserted, not even that it's "something".)

Start of Chapter 17: Understanding
Back to §17.18. Changing the meaning of pronouns
Onward to §17.20. Multiple action processing
Example 326: **Masochism Deli** Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

---

§17.20. Multiple action processing

When the player types a command like DROP ALL, this is (usually) a request to carry out more than one action. After the command parser has decided what constitutes "ALL" (a process which can be influenced using the "deciding whether all includes" activity), it forms up a list and then runs through it, starting an action for each in turn. The result usually looks something like this:

```
>GET ALL
foxglove: Taken.
snake's head fritillary: Taken.
```

However, by adding rules to the rulebook:

we can take a look at the actions intended, and rearrange or indeed change them before they take effect. To do that, we have to deal with a special list of objects. For two technical reasons this isn't stored as a "list of objects that varies" - first because it needs to exist even in low-memory situations where we can't afford full list-processing, and second because there are times when changing it might be hazardous. Instead, two phrases are provided to read the list and to write it back:

multiple object list ... list of objects

This phrase produces the current multiple object list as a value. The list will be the collection of objects found to match a plural noun like ALL in the most recent command typed by the player. If there is no multiple object, say if the command was TAKE PEAR, the list will be empty: it won't be a list of size 1.

alter the multiple object list to (list of objects)

This phrase sets the multiple object list to the given value. The list is ordinarily the collection of objects found to match a plural noun like ALL in the most recent command typed by the player, but using this phrase at the right moment (before the "generate action rule" in the turn sequence rules takes effect).

Start of Chapter 17: Understanding
Back to §17.19. Does the player mean...
Onward to §17.21. Understanding mistakes

Example 327: ★★ **The Best Till Last** Reordering multiple objects for dramatic effect.

Example 328: ★★ **Western Art History 305** Allowing EXAMINE to see multiple objects with a single command.

§17.21. Understanding mistakes

When inspiration strikes the player, he can usually be relied upon to make a good-faith effort to communicate the new idea: he will guess the right command. If he guesses wrongly, the mistake is probably the author's, because a good author will try to anticipate all possible wordings and make all of them work.

Nevertheless it is sometimes good practice to nudge the player towards the right wording - particularly if the player has the right idea but is not explicit enough: for instance, typing TALK TO JUDGE when we really want to know what is to be said (JUDGE, GUILTY); or if the player tries something like PLAY CHESS rather than MOVE PAWN TO KING 4. Similarly, if we make a casual reference such as "In your childhood days, you loved sliding in stocking feet across this hallway", a player might type SLIDE IN STOCKING FEET: a nice idea, and which deserves a nice response, even though it asks to do something beyond the scope of the story.

Inform provides a simple mechanism for recognising a command but at the same time recognising that *it does not properly specify an action*. Such commands are called "mistakes", for the sake of a memorable term, but the player has not really behaved badly, and should be helped rather than reproved. For instance:

```
Understand "act" as a mistake.
```

While that works - the command to "act" is indeed rejected - it is not very good, because no very helpful message is brought up. The following is much better:

```
Understand "act" as a mistake ("To join the actors, you have to adopt
a role in the play! Try PLAY HAMLET or similar.").
```

Or we could once again insist on a given context:

```
Understand "act" as a mistake ("To join the actors, you have to adopt
a role in the play! Try PLAY HAMLET or similar.") when the location
is the Garden Theatre.
```

That still has the drawback that the command "act hamlet" will not be recognised: so the final version we want is probably

```
Understand "act [text]" as a mistake ("To join the actors, you have to
adopt a role in the play! Try PLAY HAMLET or similar.") when the
```

```
location is the Garden Theatre.
```

since the "[text]" part will soak up any words the player types (or none), meaning that any command at all whose first word is "act" will be matched.

We need to be careful to avoid circular things like this:

```
Understand "[text]" as a mistake ("'[the topic understood]' is
something I really wish you wouldn't say.") when the topic understood
is a topic listed in table 1.
```

This doesn't work because the topic understood isn't set until the line has been understood, but Inform checks the "when..." condition before it tries to understand the line. Indeed, even this:

```
Understand "[text]" as a mistake ("'[the topic understood]' is
something I really wish you wouldn't say.").
```

is unsafe (quite apart from being unwise!) - again, "topic understood" doesn't exist for a mistake, because in a mistake, nothing is understood.

The following is often useful during beta-testing of a new work, though we would not want it in the final published edition. Many authors like to ask their testers not to try anything in particular, simply to play naturally: but to record the transcript of the session, and email it back to the author. The following command is a device to allow the tester to type a comment in to the transcript:

```
Understand "* [text]" as a mistake ("Noted.").
```

For instance, the tester might type "* DIDN'T WE SAY DARCY WAS TALL?", to which the story would reply "Noted." - and the author can search for such comments when receiving the transcript.

If we are careful, we can make the reply depend on what was typed in the mistaken command:

```
Understand "steal [something]" as a mistake ("Just TAKE [the noun]
and leave without paying: that's stealing in my book.").
```

The care comes in because Inform applies much less checking to mistakes than to other actions, and odd errors will result if we try to refer to (say) "the second noun" in

a command which did not have a second noun.

It's probably wise to take particular care if using "as a mistake" with any command which might include the mistake among what the player calls ALL: for example, if "take [sydney harbour bridge]" is understood as a mistake, then TAKE ALL will may result in this, even though the player doesn't intend any such thing.

Start of Chapter 17: Understanding

Back to §17.20. Multiple action processing

Onward to §17.22. Precedence

Example 329: ★ **Query** Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

Example 330: ★ **The Gorge at George** If the player tries to TALK TO a character, suggest alternative modes of conversation.

Example 331: ★★★ **Hot Glass Looks Like Cold Glass** Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

---

§17.22. Precedence

When several different lines of grammar are supplied to meet the same circumstances, it makes a big difference what order they are tried in. For instance, suppose we have:

```
Understand "photograph [a door]" as photographing.
```

```
Understand "photograph [an open door]" as photographing.
```

The second line is more specific than the first, so Inform takes these grammar lines the other way around: it checks for "open door" before it checks for "door". That didn't matter here, since both lines came out with the same result (the action of photographing), but it matters very much in the next example:

```
Understand "employ [a door]" as opening.
```

```
Understand "employ [an open door]" as entering.
```

More subtle is a line already seen:

```
Understand "on/in/inside" or "on top of" as "[within]".
```

Here Inform puts "on top of" before "on/in/inside", since otherwise only the "on" of "on top of" will be recognised.

Mistakes always take precedence over non-mistakes: this is intended to make sure that

```
Understand "take umbrage" as a mistake ("Nobody takes umbrage in
this story, mister.").
```

will take precedence over

```
Understand "take [something]" as taking.
```

even if there is, in fact, a character called Mr Nimbus Umbrage so that the command could conceivably make sense.

Finally, there are a few grammars where the number of values produced is different in different lines. For example, the Standard Rules include these among the possible "put" commands:

```
Understand "put [something preferably held] on" as wearing.
Understand "put [other things] on/onto [something]" as putting it on.
```

One produces a single object, the other produces two. Inform gives precedence to the first of these, that is, it tries the one with fewer values first. This is important when reading commands like "PUT MARCH ON WASHINGTON SHIRT ON", and also prevents bogus auto-completions, in which PUT HAT ON might wrongly be auto-completed as if it were PUT HAT ON THE TABLE.

Start of Chapter 17: Understanding

Back to §17.21. Understanding mistakes

Onward to Chapter 18: Activities: §18.1. What are activities?

Example 332: ★ **Some Assembly Required** Building different styles of shirt from component sleeves and collars.

Example 333: ★★★ **Lakeside Living** Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

Examples from Chapter 17: Understanding

Start of this chapter

Chapter 18: Activities

Indexes of the examples

Example Indirection *RB*

Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

In Mayan culture, colours seem to have been used as names for the primary directions: for instance, "red" implies east as the colour of sunrise. So the following might be a stylish touch for a game in which the player has to get inside the Mayan world-view:

```
"Indirection"

Understand "white" and "sac" as north. Understand
"red" and "chac" as east. Understand "yellow" and "kan"
as south. Understand "black" and "chikin" as west.
```

We could also use a colour as a verb:

```
Understand "turquoise" and "yax" as looking.
```

And now a few extra rooms to try it out in:

```
The Square Chamber is a room. "A sunken, gloomy
stone chamber, ten yards across. A shaft of sunlight
cuts in from the steps above, giving the chamber a
diffuse light, but in the shadows low lintelled doorways
to east and south lead into the deeper darkness of the
Temple."

The Wormcast is east of the Square Chamber. The
Corridor is south of the Square Chamber.

Test me with "kan / white / chac / black".
```

ExampleXYZZY

Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

RB

We have seen before how to define a new action from scratch, but we may want to review here, using a simple command that requires no objects.

```
"XYZZY"

Understand "xyzzy" or "say xyzzy" or "cast xyzzy" as
casting xyzzy.

Casting xyzzy is an action applying to nothing.

Check casting xyzzy:
    if the player does not wear the amulet of elocution,
say "You are unable to articulate the second 'z'
separately from the first, and the spell fails in a
disdainful puff. Must be Parisian magic." instead;
    if the player has the plate, say "The plate of cheeses
twitches uncomfortably, aware that it should be doing
something, but not sure what." instead.

Carry out casting xyzzy:
    move the plate to the player.

Report casting xyzzy:
    say "Under the influence of the Amulet of Elocution,
you pronounce this as Xhi-zee. And lo, from nowhere, a
[plate] appears!"

The amulet of elocution is a wearable thing. It is carried
by the player. The description is "A heavy gold ring on a
chain. If heated in an ordinary house fire, it glows with
the words, 'Moses Supposes His Toeses Are Roses.'"
```

```
The plate is a portable supporter. On the plate is a very
ripe ooze. Instead of smelling the ooze, say "It smells
like socks. This is going to be wonderful." The ooze is
edible. The printed name of the plate is "plate[if the
plate supports the ooze] of cheese[end if]". The
description of the ooze is "Definitely genuinely cheese."
Understand "cheese" as the ooze.


Instead of eating the ooze: now the ooze is nowhere;
say "You are transported..."; move the player to
Paradise.


The Cheez Factory is a room. "All around you are
squares of pressed orange polymer, or possibly cheez.
Your only hope is the magic word your uncle taught you:
XYZZY." The squares of pressed orange polymer are
scenery in the Factory. The description is "You see
nothing special about the squares of pressed orange
polymer. Nothing special at all." Understand "square" or
"cheez" as the squares.


Paradise is a room. The description is "Well, it might just
be one of the posh upper rings of purgatory, if you're
entirely honest with yourself."


Test me with "x squares / x amulet / x cheese / xyzzy /
wear amulet / xyzzy / x ooze / smell ooze / eat ooze".
```

XYZZY is a magic word from the original Adventure, and many other games respond to it with some sort of amusing message.

ExampleXylan *RB*

Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

```
If we wanted to define a brand new verb that did affect a specific object,
```

we might begin like this:

```
"Xylan"


Understand "hydrolyze [something]" as hydrolyzing.
Hydrolyzing is an action applying to one thing.


Carry out hydrolyzing:
    say "[The noun] cannot be hydrolyzed."


Instead of hydrolyzing the xylan:
    move the xylose to the holder of the xylan;
    now the xylan is nowhere;
    say "At once the xylan becomes xylose."


Plant Cell Wall is a room.


There is a xylose sample. The xylan sample is a thing in
Plant Cell Wall. The description of the xylan is "A
polysaccharide. Totally useless. If only you had some
xylose, instead!" The description of the xylose is
"Awesome!"


Test me with "x xylan / hydrolyze xylan / x xylose".
```

Of course, how our players will ever solve this problem is another question (especially if their biology and chemistry are both rusty). When adding entirely new commands to a game, it is often a good idea to provide as many ways of phrasing the command as possible; to drop hints about the correct phrasing within the game's text; or even to tell the player about the expanded command list in some documentation or help at the beginning of the game. So for instance we might also add

```
Understand "break down [something] with water" or
"break [something] down with water" as hydrolyzing.
```

And these lines will also provide syntax for our new command, without interfering with the previous syntax. It's also good to anticipate alternative (British or American) spellings. People's typing habits are hard to overcome, even if they know you are spelling the word the other way. It is probably best not to annoy them unduly. So:

```
Understand "hydrolyse [something]" as hydrolyzing.
```

Then some text in-game might offer a clue, subtle or (since this is an example) blunt:

```
Instead of examining the player, say "You're a drop of
water, which means that you can break down certain
chemicals!"
```

```
Understand "break down [something]" or "break
[something] down" as hydrolyzing.
```

And finally, we could try adding instructions explicitly:

```
Understand "help" or "hint" or "hints" or "instructions" or
"info" or "about" as asking for help. Asking for help is an
action out of world. Carry out asking for help: say "The
following commands are understood, in addition to the
standard ones: EVAPORATE, FREEZE, HYDROLYZE,
SUBLIME..."
```

```
Test more with "help / x me / break down xylan"
```

...though of course in fact these other commands won't be available until we define them, too.

This last approach, defining all the extra commands up front, is especially

useful if these commands are very technical or unusual; if they are needed early in the game, before you've a chance to educate the player; or if they are not suggested by any in-game objects. A player who encounters a tool with an obvious use, such as a hairbrush, will likely think of trying to BRUSH things with it. It's harder to rely on his guessing actions that are both outside the range of usual commands and unrelated to any of the visible props, however.

---

289 **Example** **Alpaca Farm** *RB*

A generic USE action which behaves sensibly with a range of different objects.

This example takes the ordering of grammar lines to its logical extreme, sorting the player's input into different categories depending on the kind and condition of the objects mentioned.

```
"Alpaca Farm"

Understand "use [an edible thing]" as eating.

Understand "use [a wearable thing]" as wearing.

Understand "use [a closed openable container]" as
opening. Understand "use [an open openable
container]" as closing.

Understand "use [something preferably held] on [a
locked lockable thing]" as unlocking it with (with nouns
reversed). Understand "use [something preferably held]
on [an unlocked lockable thing]" as locking it with (with
nouns reversed).
```

Understand "use [a switched off device]" as switching on.

Understand "use [something]" as using. Using is an action applying to one thing. Carry out using: say "You will have to be more specific about your intentions."

Understand "use [a door]" as opening. Understand "use [an open door]" as entering.

The Llama Pen is a room. North of the Pen is the gate. The gate is a door. North of the gate is the Rocky Path. The brown llama is an animal in the Llama Pen.

Appearance is a kind of value. The appearances are muddy, scruffy, fluffy, and dapper. The brown llama has an appearance. The brown llama is muddy. Before printing the name of the brown llama, say "[appearance] ". Before printing the name of the brown llama while grooming: say "now-[if appearance of the brown llama is less than dapper]merely-[end if]".

A grooming tool is a kind of thing. Understand "use [a grooming tool] on [something]" as grooming it with (with nouns reversed). Grooming it with is an action applying to two things. Understand "groom [something] with [something]" as grooming it with.

Carry out grooming it with:
    if the appearance of the noun is less than dapper, now the appearance of the noun is the appearance after the appearance of the noun.

Report grooming it with:

say "You attend diligently to the appearance and hygiene of [the noun]."

Instead of using a grooming tool in the presence of the brown llama:
try grooming the brown llama with the noun.

The player carries some nail nippers, a slicker brush, and an apple. The apple is edible. The brush and the nippers are grooming tools. The player wears a sombrero.

The description of the nail nippers is "Ten inches long, to give you the necessary leverage to cut tough llama toenails. It still helps to soften them up by making the llama stand in a bucket of water first, though."

The description of the slicker brush is "Fine, angled soft bristles set into a broad back, perfect for removing mud from the coat of a long-woolled llama."

The industrial-strength blower is a fixed in place device in the Llama Pen. "Attached to the nearest wall, on its own movable boom, is an industrial-strength blower for doing llama hair."

Understand "use [switched off blower]" as switching on. Understand "use [switched on blower] on [brown llama]" as grooming it with (with nouns reversed). Instead of using the blower in the presence of the brown llama, try grooming the brown llama with the blower.

Test me with "use gate / use blower / use nippers / use brush / use apple / remove sombrero / use sombrero".

Whether we actually want a USE action is a subject of some theoretical debate in the IF community. On the one hand, it helps avoid guess-the-verb problems where the player cannot figure out what term to use in order to express a fairly simple idea. On the other, it encourages the player to think that all items have one and exactly one use, rather than getting him to consider the range of possibilities that arise from having a complex vocabulary.

ExampleAnchorite *RB*

By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.

With GET DOWN, we can replace the whole command, which will not interfere with the normal function of the TAKE verb, or allow the player to attempt to GET any other directions:

```
"Anchorite"

The Solitary Place is a room. "A glittering, shimmering
desert without either locusts or honey." The pillar is an
enterable supporter in the Solitary Place. "The broken
pillar is short enough to climb and sit on." The
description of the pillar is "Once it was a monument: a
long frieze of battles and lion-hunts spirals up the side,
in honor of an earthly king." The player is on the pillar.

Understand "get down" as exiting.
```

This doesn't cover the case where the player just types "DOWN", and we don't want to preempt the normal operation of the GO action here. So instead of writing a new understand instruction, we might catch this one at the action-processing level:

```
Instead of going down when the player is on a
supporter:
```

try exiting.

Test me with "down / enter pillar / get down / down / get down".

---

291 **Example** **Cloak of Darkness** *RB*

Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

"Cloak of Darkness" is a brief example game that has been implemented in nearly every IF system currently used. It hasn't got much claim to complexity or richness, but it does exemplify many of the standard things one might want an IF language to be able to do: define descriptions and the results of actions, assign synonyms to nouns, create new verbs, handle darkness, track repeated acts, and so on.

Here is what the game looks like in Inform:

"Cloak of Darkness"

The story headline is "A basic IF demonstration."

Use scoring.

The maximum score is 2.

Whatever room we define first becomes the starting room of the game, in the absence of other instructions:

Foyer of the Opera House is a room. "You are standing in a spacious hall, splendidly decorated in red and gold,

```
with glittering chandeliers overhead. The entrance from
the street is to the north, and there are doorways south
and west."
```

```
Instead of going north in the Foyer, say "You've only just
arrived, and besides, the weather outside seems to be
getting worse."
```

We can add more rooms by specifying their relation to the first room. Unless we say otherwise, the connection will automatically be bidirectional, so "The Cloakroom is west of the Foyer" will also mean "The Foyer is east of the Cloakroom":

```
The Cloakroom is west of the Foyer. "The walls of this
small room were clearly once lined with hooks, though
now only one remains. The exit is a door to the east."
```

```
In the Cloakroom is a supporter called the small brass
hook. The hook is scenery. Understand "peg" as the
hook.
```

Inform will automatically understand any words in the object definition ("small", "brass", and "hook", in this case), but we can add extra synonyms with this sort of Understand command.

```
The description of the hook is "It's just a small brass
hook, [if something is on the hook]with [a list of things
on the hook] hanging on it[otherwise]screwed to the
wall[end if]."
```

This description is general enough that, if we were to add other hangable items to the game, they would automatically be described correctly as well.

```
The Bar is south of the Foyer. The printed name of the
bar is "Foyer Bar". The Bar is dark. "The bar, much
rougher than you'd have guessed after the opulence of
the foyer to the north, is completely empty. There seems
to be some sort of message scrawled in the sawdust on
```

```
the floor."


The scrawled message is scenery in the Bar.
Understand "floor" or "sawdust" as the message.


Neatness is a kind of value. The neatnesses are neat,
scuffed, and trampled. The message has a neatness.
The message is neat.
```

We could if we wished use a number to indicate how many times the player has stepped on the message, but Inform also makes it easy to add descriptive properties of this sort, so that the code remains readable even when the reader does not know what "the number of the message" might mean.

```
Instead of examining the message:
    increment score;
    say "The message, neatly marked in the sawdust,
reads...";
    end the story finally.
```

This second rule takes precedence over the first one whenever the message is trampled. Inform automatically applies whichever rule is most specific:

```
Instead of examining the trampled message:
    say "The message has been carelessly trampled,
making it difficult to read. You can just distinguish the
words...";
    end the story saying "You have lost".
```

This command advances the state of the message from neat to scuffed and from scuffed to trampled. We can define any kinds of value we like and advance or decrease them in this way:

```
Instead of doing something other than going in the bar
when in darkness:
    if the message is not trampled, now the neatness of
the message is the neatness after the neatness of the
```

```
    message;
        say "In the dark? You could easily disturb something."


    Instead of going nowhere from the bar when in
    darkness:
        now the message is trampled;
        say "Blundering around in the dark isn't a good idea!"
```

This defines an object which is worn at the start of play. Because we have said the player is wearing the item, Inform infers that it is clothing and can be taken off and put on again at will.

```
    The player wears a velvet cloak. The cloak can be hung
    or unhung. Understand "dark" or "black" or "satin" as the
    cloak. The description of the cloak is "A handsome
    cloak, of velvet trimmed with satin, and slightly
    splattered with raindrops. Its blackness is so deep that it
    almost seems to suck light from the room."


    Carry out taking the cloak:
        now the bar is dark.


    Carry out putting the unhung cloak on something in the
    cloakroom:
        now the cloak is hung;
        increment score.


    Carry out putting the cloak on something in the
    cloakroom:
        now the bar is lit.


    Carry out dropping the cloak in the cloakroom:
        now the bar is lit.
```

```
Instead of dropping or putting the cloak on when the
player is not in the cloakroom:
    say "This isn't the best place to leave a smart cloak
lying around."


When play begins:
    say "[paragraph break]Hurrying through the rainswept
November night, you're glad to see the bright lights of
the Opera House. It's surprising that there aren't more
people about but, hey, what do you expect in a cheap
demo game...?"


Understand "hang [something preferably held] on
[something]" as putting it on.


Test me with "s / n / w / inventory / hang cloak on hook /
e / s / read message".
```

And that's all. As always, type TEST ME to watch the scenario play itself out.

---

292 **Example The Trouble with Printing** *RB*

Making a READ command, distinct from EXAMINE, for legible objects.

```
"The Trouble with Printing"


A thing has some text called printing. The printing of a
thing is usually "blank".


Understand the command "read" as something new.
Understand "read [something]" as reading. Reading is
```

```
an action applying to one thing, requiring light. Check
reading: if the printing of the noun is "blank", say
"Nothing is written on [the noun]." instead. Carry out
reading: say "You read: [printing of the noun][line
break]". Report reading: do nothing.


The Archive is a room.


Berkeley's report is a thing in the Archive. The
description is "A report from Governor Sir William
Berkeley of Virginia, in 1671, in answer to the queries
sent by the Commissioners of Plantations the year
previous. Of this report the better part is burned and
only a tail fragment remains." The printing of Berkeley's
report is "I thank God, [italic type]there are no free
schools[roman type] nor [italic type]printing[roman type],
and I hope we shall not have these hundred years; for
[italic type]learning[roman type] has brought
disobedience, and heresy, and sects into the world, and
[italic type]printing[roman type] has divulged them..."


Test me with "examine report / read report".
```

Since we defined reading as an action requiring light, we could further distinguish reading and examining (if we wanted) by writing some different visibility rules for it.

ExampleLanista 2 *RB*

Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an ATTACK IT WITH action is created to replace regular attacking. Also folds a new DIAGNOSE command into the system.

Back in the chapter on randomization, we explored a way to create a randomized combat system. That system didn't allow for multiple weapons, though. Here we explore how to create an ATTACK IT WITH action that will let the player choose between weapons with different maximum powers.

We're also going to rewrite that original "instead of attacking:" rule into an attacking it with action that can be performed equally by the player or by any of the player's enemies.

```
"Lanista, Part Two"


The Arena is a room. "Sand, blood, iron. These festivals
are normally held on hot days, but the sun has gone
behind a cloud and fat drops of rain now and then
spatter the arena floor." The gladiator is a man in the
Arena. "A bare-chested Scythian gladiator faces you,
wielding [a list of weapons carried by the gladiator]."


Section 1 - Hit Points


A person has a number called maximum hit points. A
person has a number called current hit points.


The maximum hit points of the player is 35. The
maximum hit points of the gladiator is 25.
```

In our simpler version of this example we set the current hit points by hand, but in a game with many characters this would get dull and repetitive, so here we'll use a "when play begins" to set all current hit point values automatically to maximum:

```
When play begins:
    repeat with victim running through people:
        now the current hit points of the victim is the
maximum hit points of the victim.


Definition: a person is dead if his current hit points are
less than 0.
```

Section 2 - Diagnosis

Diagnosing is an action applying to one visible thing. Understand "diagnose [something]" as diagnosing.

Check diagnosing:
    if the noun is not a person, say "Only people can have diagnoses." instead.

Carry out diagnosing:
    say "[if the noun is the player]You have[otherwise][The noun] has[end if] [current hit points of the noun] out of a possible [maximum hit points of the noun] hit points remaining."

Section 3 - Weapons

A weapon is a kind of thing. A weapon has a number called the maximum damage. The maximum damage of a weapon is usually 4.

The gladiator carries a weapon called a trident. The maximum damage of the trident is 5. The gladiator carries a weapon called a net. The maximum damage of the net is 1.

The player carries a weapon called a mace. The maximum damage of the mace is 3.

Section 4 - Attacking it with

In our new system, we want to specify what is being used for an attack. This means that we need to create a new "attacking it with" action, and also that we should disable the existing "attacking..." command.

Here's why: If we leave the default attack command in place, Inform will continue to accept commands like >ATTACK GLADIATOR, but reply foolishly with the default "Violence is not the answer..." response.

A somewhat better approach would be to change the reply of >ATTACK GLADIATOR to say something like "You must specify a weapon to attack with." But this is still less than ideal, because it means that the player has to then rewrite his entire command. If, on the other hand, we take out "ATTACK GLADIATOR" entirely, the game will always prompt "What do you want to attack the gladiator with?" -- which teaches the player the correct command structure for this particular game, and avoids pretending to understand any command that is not meaningful within this game.

This is a little bit of work because ATTACK has a lot of synonyms in the default library, but if we look through the actions index we can find them all:

```
Understand the commands "attack" and "punch" and
"destroy" and "kill" and "murder" and "hit" and "thump"
and "break" and "smash" and "torture" and "wreck" as
something new.
```

Now we make our new command:

```
Attacking it with is an action applying to one visible thing
and one carried thing. Understand "attack [someone]
with [something preferably held]" as attacking it with.
```

Note that we've specified "one carried thing", because we want the player to pick up a weapon to use if necessary. And now we assign all the old attack vocabulary to apply to the new command:

```
Understand the commands "punch" and "destroy" and
"kill" and "murder" and "hit" and "thump" and "break"
and "smash" and "torture" and "wreck" as "attack".
```

This may seem counter-intuitive, but order of source code matters here: we first get rid of the old, default vocabulary, then define our new action, then make the vocabulary apply to that new action. Inform will now understand >HIT GLADIATOR WITH TRIDENT, >BREAK GLADIATOR WITH TRIDENT, and so on.

Our new action is also a perfect place to use an action variable: we're going to need to choose an amount of damage done and refer to that several times in our action rules. So let's set that up first:

```
The attacking it with action has a number called the
damage inflicted.


Setting action variables for attacking something with
something:
    if the second noun is a weapon:
        let the maximum attack be the maximum damage
of the second noun;
        now the damage inflicted is a random number
between 1 and the maximum attack.


Check an actor attacking something with something
(this is the can't attack with something that isn't a
weapon rule):
    if the second noun is not a weapon:
        if the actor is the player, say "[The second noun]
does not qualify as a weapon.";
        stop the action.


Check an actor attacking something with something
(this is the can't attack a non-person rule):
    if the noun is not a person:
        if the actor is the player, say "[The noun] has no life
to lose.";
        stop the action.


Carry out an actor attacking something with something
(this is the standard attacking it with a weapon rule):
    decrease the current hit points of the noun by the
```

```
damage inflicted;
    if the noun is dead:
        now the noun is nowhere.
```

Though our checks and carry-out rules are similar regardless of who is acting, we're going to want actions to be described differently for different actors, so we'll use separate "report attacking" and "report someone attacking" rules. We'll also make some special cases for when the character has died as a result of the attack:

```
Report attacking a dead person with something (this is
the death-report priority rule):
    say "You attack with [the second noun], killing [the
noun]!" instead.
```

```
Report attacking someone with something (this is the
normal attacking report rule):
    say "You attack [the noun] with [the second noun],
causing [damage inflicted] point[s] of damage!" instead.
```

```
Report someone attacking the player with something
when the player is dead (this is the player's-death
priority rule):
    say "[The actor] attacks you with [the second noun],
finishing you off!";
    end the story;
    stop the action
```

```
Report someone attacking the player with something
(this is the standard report someone attacking the player
with rule):
    say "[The actor] attacks you with [the second noun],
causing [damage inflicted] point[s] of damage!" instead.
```

```
Report someone attacking something with something
(this is the standard report attacking it with rule):
    say "[The actor] attacks [the noun] with [the second
noun], causing [damage inflicted] point[s] of damage!"
instead.
```

```
When play begins:
    now the left hand status line is "You: [current hit
points of player]";
    now the right hand status line is "Gladiator: [current
hit points of gladiator]".



Every turn (this is the gladiator-attack rule):
    if the gladiator is not dead, try the gladiator attacking
the player with a random weapon which is carried by the
gladiator.



Test me with "hit gladiator with mace / kill gladiator /
drop mace / attack gladiator / attack gladiator with mace
/ g / g".
```

Those devoted to role-playing will note that our form of randomization is still pretty naive: most RPG systems use multiple dice in order to create more interesting probability curves. For a system that simulates actual dice-rolling, see the full "Reliques of Tolti-Aph" game.

Example Shawn's Bad Day *RB*

Allowing the player to EXAMINE ALL.

We can add the handling of multiple objects to an existing action simply by adding in a line of grammar using "[things]". In response, Inform will consider every object accepted by the token, and perform the action once for each of those objects. Thus:

```
"Shawn's Bad Day"



The Treasury is a room. The vault is a lockable locked
closed openable container in the Treasury. It is fixed in
place. "A massive vault fills up one wall." The
```

description is "The vault's system includes [a list of things which are part of the vault]."

A little green light, a little blue light, a little red light, a thin black pane of glass, a laser beam, a retinal scanner, a thumbprint ID plate, a dial, and a large lever are part of the vault.

The security guard is a man in the Treasury. The description is "His name is Shawn, and he doesn't look happy."

The description of the green light is "Off." The description of the blue light is "Tranquilly on." The description of the red light is "Angrily flashing."

Understand "examine [things]" as examining.

Test me with "examine all".

Example **The Left Hand of Autumn** *RB*

The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple examination command that describes groups in special ways.

Suppose that we have a game in which groups of objects can have meaning apart from their individual significance -- perhaps there are spells that can only be cast by collecting just the right items in the same place.

In this case, one of the things the player might like to be able to do is look at several items together and get a special response, different from

looking at the items individually.

To make this happen, we need to do several things:

(1) we need to create a version of the EXAMINE command that can apply to multiple objects at once.
(2) we need to correct the way Inform normally deals with multiple-object commands, because we want our group description to print only one time, and we want to avoid stubs such as "pear: ... apple: ..." before or after the group description.
(3) we need to define a way for Inform to identify interesting groups and describe them.

"The Left Hand of Autumn"

Section 1 - Procedure

Understand "examine [things]" or "look at [things]" as multiply-examining. Multiply-examining is an action applying to one thing.

Understand "examine [things inside] in/on [something]" or "look at [things inside] in/on [something]" as multiply-examining it from. Multiply-examining it from is an action applying to two things.

Group-description-complete is a truth state that varies.

Carry out multiply-examining it from:
    try multiply-examining the noun instead.

```
Check multiply-examining when group-description-
complete is true:
    stop the action.


Carry out multiply-examining:
    let L be the list of matched things;
    if the number of entries in L is 0, try examining the
noun instead;
    if the number of entries in L is 1, try examining entry 1
of L instead;
    describe L;
    say line break;
    now group-description-complete is true.


Before reading a command:
    now group-description-complete is false.
```

Now for step 2, overriding Inform's usual output of names of objects:

```
The silently announce items from multiple object lists
rule is listed instead of the announce items from multiple
object lists rule in the action-processing rules.


This is the silently announce items from multiple object
lists rule:
    unless multiply-examining or multiply-examining
something from something:
        if the current item from the multiple object list is not
nothing, say "[current item from the multiple object list]:
[run paragraph on]".


Definition: a thing is matched if it is listed in the multiple
object list.
```

We'll save our "to describe" phrase until Section 2, when we can give the game specific instructions about how to report different lists of objects.

Now, the player might also want to be able to refer to a group of item by some kind of group name, so let's add the option of creating a Table of

Collective Names which will interpret these:

```
After reading a command:
    repeat through the Table of Collective Names:
        let N be "[the player's command]";
        let Y be relevant list entry;
        while N matches the regular expression "[name-
text entry]":
            replace the regular expression "(.*)[name-text
entry](.*)" in N with "\1[Y]\2";
        change the text of the player's command to N.


Report taking something:
    say "You pick up [the noun]." instead.
```

And as a bit of polish, because we'd like SEARCH TABLE to have the same effect as EXAMINE ALL ON TABLE:

```
Understand "look on [something]" as searching.


Instead of searching something which supports at least
two things:
    let L be the list of things supported by the noun;
    describe L.
Instead of searching something which contains at least
two things:
    let L be the list of things contained by the noun;
    describe L.


Section 2 - Scenario


Eight-Walled Chamber is a room. "A perfectly octagonal
room whose walls are tinted in various hues."
```

```
The display table is a supporter in the Chamber. A twig
of rowan wood is on the table.


The player carries an apple and a pear.


A glove is a kind of thing. A glove is always wearable.
Understand "glove" as a glove. The player carries a left
glove and a right glove. The left glove and the right
glove are gloves.
```

Now we define a few actual lists of items:

```
Fruit list is a list of objects which varies. Fruit list is {
apple, pear }.
Glove list is a list of objects which varies. Glove list is {
right glove, left glove }.
Arcane list is a list of objects which varies. Arcane list is
{ left glove, twig, pear }.


To describe (L - a list of objects):
    sort L;
    if L is fruit list:
        say "Just a couple of fruits.";
    otherwise if L is glove list:
        say "It's a matched pair of fuzzy blue gloves.";
    otherwise if L is arcane list:
        say "To anyone else it might look like a random
collection of objects, but these three things -- [L with
definite articles] -- constitute a mystic key known as the
Left Hand of Autumn. They practically hum with power.";
    otherwise:
        say "You see [L with indefinite articles]."


When play begins:
    sort fruit list;
    sort glove list;
    sort the arcane list.
```

We sort the lists so that regardless of how we change the rest of the

code (and the order in which objects are coded), the resulting list will always be in sorted order and ready to compare with the list of items the player wants to look at. And thanks to the "Reading a command" code we wrote earlier, we can also teach the game to understand the player's references to "the left hand of autumn" as a specific collection of items.

```
Table of Collective Names

relevant list
"[left hand of autumn]"
"[gloves]"
"[glove of ice]"
```

```
Test me with "x apple and pear / x left and right / put
pear on table / put left glove on table / x all on table / put
all on table / examine all on table / get apple, twig, pear /
x all on table / search table".
```

ExampleIsh. *RB*

A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

```
"Ish."
```

```
Ichiro's Dubious Sushi Hut is a room. "Despite the allure
of the dusty plastic sushi models in the window, you're
beginning to have second thoughts about the selection
of this particular restaurant for your rendezvous with
Agent Fowler. There are no other patrons, for one thing.
Afternoon sunlight filters lazily through the window and
illuminates a number of empty glass-topped tables, at
each of which is a chopstick dispenser (in form of cute
ceramic cat) and a pitcher of soy sauce (sticky).
```

```
The sushi bar itself is what gives the most pause,
however. Behind it sits an angry-looking Japanese
woman, aggressively eating a Quarter Pounder with
Cheese."
```

We can, when necessary, accept any text at all as a token:

```
Understand "help [text]" or "help about [text]" as getting
help about. Understand the commands "instructions" or
"hint" or "hints" or "menu" or "info" or "about" as "help".


Getting help about is an action applying to one topic.
```

After that, we can use "the topic understood" to refer to the text we read:

```
Carry out getting help about:
    if the topic understood is a topic listed in the Table of
Standard Help:
        say "[explanation entry][paragraph break]";
    otherwise:
        say "You're out of ideas."


Table of Standard Help

topic                explanation
"sushi"              "Sushi is just vinegar rice that sushi inevitably entails raw fish,
                     but it is in fact just rice with rice vinegar on it. It's just that the really
                     good kinds have raw fish in."
"wasabi"             "Sushi bars usually have wasabi in the middle, and as long as
                     you don't go too crazy with the wasabi, all should be well."
"California roll"    "The California roll is a concept"
"raw fish"           "Japan's having this unusual delicacy is near zero."
"Fowler"             "We'll have time enough to deliver later. If Fowler ever turns up.
                     Where is she, anyway?"
```

Since the player may not know what all the help options are, we might as well let him get an overview, as well.

```
Understand "help" as summoning help. Summoning
help is an action applying to nothing.


Carry out summoning help:
    say "Help is available about the following topics.
Typing HELP followed by the name of a topic will give
further information.[paragraph break]";
    repeat through the Table of Standard Help:
        say " [title entry]: [summary entry][line break]".


Test me with "help / help about microdot / help
cucumber / help california roll".
```

---

297 **ExampleNameless** *RB*

ASKing someone about an object rather than about a topic.

By default, ASK SOMEONE ABOUT... applies only to a text token. We might want also to offer the player the option of asking characters about pieces of physical evidence. This example implements an ASK PERSON ABOUT THING command that is mostly synonymous with SHOW, with the added nuance that the player can ask about things that are not currently visible, as long as he has encountered them at some time in the past.

```
"Nameless"


The Black Chamber is a room. "Despite its menacing
name, it is quite an ordinary room, underlying the post
office above. Here letters are brought each day,
unsealed, transcribed, resealed, and sent again on their
way; their contents then analyzed and recorded."


The Nameless Advisor is a woman in the Black
```

```
Chamber. "A woman whose name has never been
disclosed to you sits at the window, writing numbers on
a sheet of paper." The Advisor carries a sheet of paper.
Understand "woman" as the nameless advisor.
```

```
The player carries a letter from the emperor. The
description of the letter is "Though its origin is obvious,
its meaning is secret: the letters are an inexplicable
jumble."
```

Now we create our new action, "interrogating it about". We write the grammar lines so that we can show any object in sight to someone, but also ask someone about any object that we have ever interacted with in the game, whether it is currently visible or not.

```
A thing can be known or unknown. The Nameless
Advisor is known.
```

```
Understand "ask [someone] about [any known thing]" as
interrogating it about. interrogating it about is an action
applying to two visible things.
```

Now we replace and redirect the showing action. This gets rid of the requirement in the default library that the player be holding anything he shows to another character:

```
Understand the commands "show" and "display" and
"present" as something new.
```

```
Understand "show [something] to [someone]" or "display
[something] to [someone]" or "present [something] to
[someone]" as interrogating it about (with nouns
reversed). Understand "show [someone] [something]"
as interrogating it about.
```

This bit keeps track of what the player has seen, for the purposes of "any known thing":

```
Before printing the name of something (called the
target): now the target is known.
```

Here we define what happens by default when we interrogate someone about something; we use the same response we get to asking someone about something that isn't otherwise interesting:

```
Carry out interrogating someone about something:
    say "There is no reply."
```

Now redirect all asking to a topic table, and all interrogating to an object table:

```
Instead of asking Nameless Advisor about a topic listed
in the Table of Nameless Advisor Topics:
    say "[reply entry][paragraph break]".
```

```
Instead of interrogating Nameless Advisor about an item
listed in the Table of Nameless Advisor Items:
    say "[reply entry][paragraph break]".
```

```
Table of Nameless Advisor Items

item                reply
"letter"            "'Letter enciphered,' she remarks[if the advisor can see the letter], glancing over the contents[otherwise], after you have offered a detailed description[end if]. 'A substitution cipher of some complexity, I believe.'"
"Advisor"           "She listens to your inquiries about her identity and parentage with a placid smile, but does not answer."
```

```
Table of Nameless Advisor Topics

topic               reply
"cipher"            "'I know many dozens of ciphers,' she replies, smiling in a
```

disquieting way."

"One letter is allowed to stand for another,' she explains, folding her hands together patiently. The backs of both hands are tattooed with silvery stars."

And just so that we can test what happens when asking someone about something out of sight:

```
The safe box is a container in the Chamber. It is fixed in
place. It is openable and closed.
```

...and something unknown:

```
The poisonous apple is a thing.
```

```
Test me with "test sight / test knowledge".
```

```
Test sight with "i / x letter / ask Nameless Advisor about
cipher / show cipher to Nameless Advisor / ask
Nameless Advisor about the letter / show the letter to
Nameless Advisor / show Nameless Advisor the letter".
```

```
Test knowledge with "open safe box / put letter in safe
box / close safe box / ask Nameless Advisor about the
letter / show the letter to Nameless Advisor / ask
Nameless Advisor about the apple".
```

ExampleSafety *RB*

A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.

"Safety"

The Vault is a room. "Snug yet paranoid, this represents the state of the art in cheerless security." The Safe is here. "A mammoth safe, with a dial which can spin to any number, has pride of place. It must weigh about the same as a small car, so don't get any ideas." Instead of opening the safe, say "The safe opens only when turned to the correct combination."

In the Safe is a silver florin. The Safe is closed and fixed in place. Understand "dial" as the Safe.

Spinning it to is an action applying to one thing and one number. Check spinning it to: if the noun is not the Safe, say "[The noun] does not spin." instead. Report spinning it to: say "Click! and nothing else happens."

Understand "spin [something] to [a number]" as spinning it to.

After spinning the closed Safe to 1384: now the Safe is open; say "Clonk! and the safe door swings slowly open, revealing [a list of things in the Safe]."

Test me with "open safe / spin safe to 1131 / open safe / spin safe to 1384 / x safe / get florin".

---

299 **Example** **Tom's Midnight Garden** *RB*

A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running. Time on the face is also reported differently depending on whether the clock is analog or digital.

Time can also be understood as a token, and the time parsed will be recorded as "the time understood". So therefore, if we wish for clocks which may be set:

```
"Tom's Midnight Garden"

A clock is a kind of device. A clock has a time called the
current time. A clock can be analog or digital. The
current time of a clock is usually 9:01 AM. The
description of a clock is "It shows the time to be [if
analog]about [the current time to the nearest five
minutes in words][otherwise][the current time][end if]."

Understand "set [clock] to [time]" as setting it by time.
Setting it by time is an action applying to one thing and
one time.

Instead of setting a clock to something:
    say "[The noun] can be set only to a time of day, such
as 8:00 AM, or midnight."

Carry out setting a clock by time:
    now the current time of the noun is the time
understood.

Report setting a clock by time:
    say "You set [the noun] to [time understood]."

Every turn:
    repeat with item running through switched on clocks:
        now the current time of the item is one minute after
the current time of the item.
```

```
The Hall is a room. The grandfather clock is a fixed in
place analog clock in the Hall. The travel clock is a
switched on digital clock in the Hall. When play begins:
now the right hand status line is "[time of day]".


Test me with "examine grandfather clock / set it to
midnight / switch it on / wait / wait / wait / examine it / set
travel clock to 4:12 / examine it".
```

---

300 **Example** **Ibid.** *RB*

A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

"Hitchhiker's Guide to the Galaxy" introduced the idea of footnoted descriptions, and various IF games since have toyed with the idea. The recommended implementation in Inform 6 involved keeping an assortment of footnote objects around, but in Inform 7 the table is a much tidier way of handling the same problem.

```
"Ibid."


The Ship Inn is a room. "Here you are in a lovely pub
which your guidebook assures you is extremely
authentic. [1 as a footnote].


To your left sits a party of Italians, with their guidebook.


To your right is a silent, but not unappealing, young
man.".
```

```
A party of Italians and a silent young man are people in
the Ship Inn. The Italians and the young man are
scenery.


The table is a supporter in the Ship Inn. On the table is
a mysterious pie. The description of the pie is "Your
waitress told you it was the specialty of the day, Steak
and Owl Pie. [2 as a footnote]." The pie is edible.


Table of Footnotes

assignment
"Francis Drake ate here, if the sign on the door is to be believed"
"this is unlikely, considering that owls are protected animals in
England these days [3 as a footnote]"
"moreover, you can't imagine that owl would be very tasty"


Footnotes mentioned is a number that varies.
```

Whenever we mention a footnote for the first time, we need to assign it a number, which we will use consistently thereafter. And it's probably a good idea to protect ourselves against the author accidentally using a number too large for the footnote table, too. So:

```
To say (footnote - a number) as a footnote:
    if footnote > number of filled rows in the Table of
Footnotes:
        say "Programming error: footnote assignment out
of range.";
    otherwise:
        choose row footnote in the Table of Footnotes;
        if there is an assignment entry:
            say "([assignment entry])";
        otherwise:
            increment footnotes mentioned;
            choose row footnote in the Table of Footnotes;
            now assignment entry is footnotes mentioned;
            say "([assignment entry])".
```

Now, in order to let the player view these footnotes, we'll need to parse numbers.

```
Understand "footnote [number]" as looking up a
footnote.


Looking up a footnote is an action applying to one
number.


Check looking up a footnote:
    if the number understood > footnotes mentioned, say
"You haven't seen any such footnote." instead;
    if the number understood < 1, say "Footnotes are
numbered from 1."


Carry out looking up a footnote:
    choose row with assignment of number understood in
the Table of Footnotes;
    say "([assignment entry]): [note entry]."


Test me with "footnote 1 / examine pie / footnote 2 /
footnote 3".
```

This method does require us to keep track of where a footnote appears in the table. If we found this inconvenient, we could add a column to the footnote table so that we could invoke it with tags like "[appearance quip as a footnote]".

---

301 Example One of Those Mornings *RB*

A FIND command that allows the player to find a lost object anywhere

Suppose that, contrary to the usual rules of interactive fiction, we want to allow the player to discover the locations of things he hasn't actually seen

yet:

```
"One of Those Mornings"

Understand "find [any thing]" as finding.

Finding is an action applying to one visible thing.

Carry out finding:
    if the player is carrying the noun:
        say "You're holding [the noun]!";
    otherwise:
        say "You left [the noun] [if the noun is on a
supporter]on[otherwise]in[end if] [the holder of the
noun]."
```

The holder of the noun can be a room, a supporter, or a container: the phrase is not picky. We would want to be a little more careful if it were ever possible for an item to have been "removed from play" in our game, since then the holder could be nothing, and that would have odd results. In this particular example, though, that will not arise.

And that's it, as far as the find command goes. The rest is local color.

```
The Exhibition Room is a room. It contains a closed
locked lockable transparent openable container called
the display case. The display case contains a priceless
pearl. The display case is scenery. The description of
the Exhibition Room is "By far the finest thing in the
room is a priceless pearl in a glass display case. It
should of course be yours[if key is not visible], if only
you can remember where you hid the key[end if]."

The silver key unlocks the display case.
```

```
A jade vase, a teak chest, a bronze teakettle, and a
child's burial casket are openable closed containers in
the Exhibition Room.


After taking the pearl:
    say "The pearl rolls into your hand, gleaming in the
oblique light; your fortune is made.";
    end the story finally.
```

If we want to have the key found in different places when the game is replayed:

```
When play begins:
    let the space be a random container which is not the
display case;
    move the silver key to the space.


Every turn:
    say "Your watch ticks with maddening loudness."


The time of day is 1:02 AM.


At 1:08 AM: say "The security guard arrives to find you
fumbling about with keys. Curses."; end the story.


Test me with "find pearl / find teakettle / get teakettle /
find teakettle / find key".
```

A FOLLOW command allowing the player to pursue a person who has just left the room.

Suppose we want the player to be able to go after characters who are moving around the map. The trick, of course, is that once characters are gone they are no longer visible to "follow [person]", so we need "follow [any person]" to find them.

```
"Actaeon"

A person has a room called last location.

Understand "follow [any person]" as following.
Understand the commands "chase" and "pursue" as
"follow".

Following is an action applying to one visible thing.

Check following:
    if the noun is the player, say "Wherever you go, there
you are." instead;
    if the noun is visible, say "[The noun] is right here."
instead;
    if the last location of the noun is not the location, say
"It's not clear where [the noun] has gone." instead.
```

Here again the best route comes in handy:

```
Carry out following:
    let the destination be the location of the noun;
    if the destination is not a room, say "[The noun] isn't
anywhere you can follow." instead;
    let aim be the best route from the location to the
destination;
    say "(heading [aim])[line break]";
    try going aim.
```

```
Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.


Artemis is a woman in Corinth.
```

We do also have to make sure that whenever we move a person from room to room, we record where they were moved from; otherwise, our clever restrictions about whom the player can pursue will not work properly.

```
To move (pawn - a person) tidily to (target - a room):
    now the last location of the pawn is the holder of the
pawn;
    move the pawn to the target.


Every turn:
    let current location be the location of Artemis;
    let next location be a random room which is adjacent
to the current location;
    if Artemis is visible, say "Artemis heads to [the next
location].";
    move Artemis tidily to next location;
    if Artemis is visible, say "Artemis arrives from [the
current location]."


Test me with "wait / follow artemis / follow artemis /
follow artemis".
```

A book with pages that can be read by number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

Suppose we have a book that the player must consult page-by-page, and we want to be able to accept all of the following input:

```
> READ BOOK (to choose a random page and read it)
> READ PAGE 1 IN BOOK
> READ PAGE 2
> READ THE LAST PAGE OF THE BOOK
> READ THE NEXT PAGE
> READ PREVIOUS PAGE IN BOOK
> READ THE FIRST PAGE
```

One approach would be to write many different understand rules and actions: one action for reading randomly, one for reading a specific page, one for reading the first page, one for reading the previous page, one for reading the next page, and one for reading the last page. But this gets tedious to construct and maintain.

More usefully, we could consider that all of the last four options are essentially the same action at heart: the player is asking to read a page in the book using a name rather than a number, and we will have to perform a minor calculation to discover what the number should be. Here's an implementation using named values to resolve this problem:

```
"Pages"


The Library is a room. The sinister book is carried by the
player. The sinister book has a number called the last
page read. The sinister book has a number called the
length. The length of the sinister book is 50.


Understand the command "read" as something new.


Understand "read [something]" or "consult [something]"
or "read in/from [something]" as reading. Reading is an
action applying to one thing, requiring light.
```

Understand "read [number] in/from/of [something]" or "read page [number] in/from/of [something]" or "look up page [number] in/from/of [something]" or "consult page [number] in/from/of [something]" as reading it in. Reading it in is an action applying to one number and one thing, requiring light.

Named page is a kind of value. The named pages are first page, last page, next page, previous page.

To decide what number is the effective value of (L - last page):
    decide on the length of the book.

To decide what number is the effective value of (F - first page):
    decide on 1.

To decide what number is the effective value of (N - next page):
    let X be the last page read of the book plus 1;
    decide on X.

To decide what number is the effective value of (P - previous page):
    let X be the last page read of the book minus 1;
    decide on X.

Understand "read [named page] in/from/of [something]" or "read the [named page] in/from/of [something]" as reading it relatively in. Reading it relatively in is an action applying to one named page and one thing, requiring light.

Does the player mean reading something in the sinister book: it is very likely.

This is the book requirement rule:
    if the player is not carrying the sinister book, say "You're not reading anything." instead.

Check reading it relatively in:
    if the second noun is not the sinister book, say "There are no pages in [the second noun]." instead;
    abide by the book requirement rule.

Carry out reading it relatively in:
    let N be the effective value of the named page understood;
    now the number understood is N;
    try reading N in the book.

Check reading it in:
    if the second noun is not the sinister book, say "There are no pages in [the second noun]." instead;
    abide by the book requirement rule.

Check reading it in:
    if the number understood is greater than the length of the sinister book, say "There are only [length of sinister book in words] pages in the book." instead;
    if the number understood is less than 1, say "The page numbering begins with 1." instead.

Carry out reading it in:
    read page number understood.

Check reading:

if the noun is not the sinister book, say "There are no pages in [the noun]." instead;
abide by the book requirement rule.

Carry out reading:
let N be a random number between 1 and the length of the sinister book; now the number understood is N;
say "You flip the pages randomly and arrive at page [the number understood]:[paragraph break]";
try reading the number understood in the sinister book.

Table of Book Contents

page	content
2	dhuma jyotih salila marutam / samnipatah kva meghah / samdes
	arthah kva patukaranaih / pranibhih prapaniyah"
13	mathesteron pws eipe kai saphesteron"
50	rgshilz maplot..."

To read page (N - a number):
now the last page read of the sinister book is N;
if there is a content corresponding to a page of N in the Table of Book Contents:
choose row with a page of N in the Table of Book Contents;
say "You read: '[content entry]'[paragraph break]";
otherwise:
say "Page [N] appears to be blank."

To read page (N - 47):
say "Your eyes burn; your ears ring. Beneath your gaze, the dreadful sigils writhe, reminding you of that which lies outside the edges of the universe...";
end the story saying "You have lost your remaining sanity".

Test me with "read from the sinister book / read the

book / read the next page / read page 2 / read previous page / g / read the first page / read the last page of the book / read the next page / read 47 in book".

Example **Down in Oodville** *RB*

Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

Now and then in IF there is a situation where we need to ask the player for a numbered choice rather than an ordinary action command. What's more, that numbered choice might change during the game, so we don't want to just hard-wire the meanings of "1", "2", and "3" whenever the player types them.

A better trick is to keep a list or table (we'll use a table here because it involves slightly less overhead) recording what the player's numerical choices currently mean. Then every time the player selects a number, the table is consulted, and if the number corresponds to something, the player's choice is acted on.

In our example, we'll have a transporter pad that can take the player to any room in the game that he's already visited. (Just for the sake of example, we'll start him off with a few pre-visited rooms.)

"Down in Oodville"

Section 1 - Method

Understand "[number]" as selecting.

Selecting is an action applying to one number.

Check selecting: [assuming we don't want to be able to

```
transport from just anywhere]
    if the player is not on the transporter pad:
        say "You can transport only from the transporter
pad. From other places than the transporter room, you
can HOME to your base ship, but not leap sideways to
other locations.";
        empty the transport options instead.



Check selecting:
    if the number understood is greater than the number
of filled rows in the Table of Transport Options or the
number understood is less than one:
        say "[The number understood] is not a valid option.
";
        list the transport options instead.



Carry out selecting:
    let N be the number understood; [not actually a
necessary step, but it makes the next line easier to
understand]
    choose row N in the Table of Transport Options;
    if the transport entry is a room:
        move the player to the transport entry;
    otherwise:
        say "*** BUG: Improperly filled table of transport
options ***" [It should not be possible for this to occur,
but we add an error message for it so that, if it ever
does, we will know what is causing the programming
error in our code]



To list the transport options:
    let N be 1;
    say "From here you could choose to go to: [line
break]";
    repeat through the Table of Transport Options:
        say " [N]: [transport entry][line break]";
        increment N.



To empty the transport options:
    repeat through the Table of Transport Options:
        blank out the whole row; [first we empty the table]
```

To load the transport options:
    repeat with interesting room running through visited rooms which are not the Transporter Room:
        choose a blank row in the Table of Transport Options;
        now the transport entry is the interesting room.

Table of Transport Options
transport
an object
with 3 blank rows. [In the current scenario, the number of blank rows need never be greater than the number of rooms in the game, minus the transport room itself.]

Understand "home" as homing. Homing is an action applying to nothing.

Check homing:
    if the player is in the Transporter Room:
        say "You're already here!" instead.

Carry out homing:
    move the player to the transporter room.

Section 2 - Scenario

The Transporter Room is a room.

Oodville is a visited room.

```
Midnight is a visited room. The Diamond City is west of
Midnight.



The transporter pad is an enterable supporter in the
Transporter Room. "The transporter pad in the middle of
the floor is currently dull blue: powered but unoccupied."



After entering the transporter pad:
    say "The transporter beeps and glows amber as you
step onto its surface. A moment later a hologram
displays your options. [run paragraph on]";
    empty the transport options;
    load the transport options;
    list the transport options.



Test me with "get on pad / 0 / -1 / 8 / 2 / look / w / home
/ get on pad / get off pad / 3".
```

If we wanted to replace the regular command structure entirely with numbered menus, or use menus to hold conversation options, we could: several Inform extensions provide these functions.

ExampleStraw Into Gold *RB*

Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or "man" -- depending on which the player last used -- until the first time the player refers to him as "Rumpelstiltskin".

```
"Straw Into Gold"



The Cell is a room. Rumpelstiltskin is an improper-
named man in the Cell. Rumpelstiltskin can be identified
or unidentified. Rumpelstiltskin is unidentified.
```

```
R-name is a kind of value. The R-names are dwarf, guy,
dude, and man-thing. Rumpelstiltskin has an R-name.
Understand "[R-name]" as Rumpelstiltskin.
```

Our example is slightly complicated by the fact that "man" is a name already known to Inform, so we can't re-use it as a kind of value. This is possible to work around, though:

```
Understand "man" as man-thing.
```

Now we borrow from the Activities chapter to look at the exact wording of the player's command:

```
After reading a command:
    if the player's command includes "[R-name]",
        now the R-name of Rumpelstiltskin is the R-name
understood;
    if the player's command includes "Rumpelstiltskin":
        now Rumpelstiltskin is identified;
        now Rumpelstiltskin is proper-named.


Rule for printing the name of Rumpelstiltskin when
Rumpelstiltskin is unidentified:
    if the R-name of Rumpelstiltskin is man-thing:
        say "man";
    otherwise:
        say "[R-name]".


Test me with "x dwarf / x guy / x dude / look / x
rumpelstiltskin / look / x man".
```

ExampleMisadventure *RB*

A going by name command which does respect movement rules, and accepts names of rooms as commands.

The original Adventure allowed the player to type the names of rooms in order to move to them, and it is now not too difficult for us to do the same. Adventure restricted this option to adjacent rooms, but we might want to be a bit more flexible, so we will accept any room:

```
"Misadventure"

Plover Room is a room. "You're in a small chamber lit by an eerie green light. An extremely narrow tunnel exits to the west. A dark corridor leads northeast."

The Dark Corridor is northeast of Plover Room. Plover Room is south of the Dark Corridor. The printed name of the Dark Corridor is "Dark Room". The description of the Dark Corridor is "You're in the dark-room. A corridor leading south is the only exit."

The Alcove is west of Plover Room. "You are in an alcove. A small northwest path seems to widen after a short distance. An extremely tight tunnel leads east. It looks like a very tight squeeze. An eerie light can be seen at the other end."

Northwest of the Alcove is the Misty Cavern. The description of Misty Cavern is "You are following a wide path around the outer edge of a large cavern. Far below, through a heavy white mist, strange splashing noises can be heard. The mist rises up through a fissure in the ceiling. The path exits to the south and west." West of Misty Cavern is the Alcove.

Understand "[any room]" as going by name. Understand "go to [any room]" as going by name.
```

```
Going by name is an action applying to one thing.
```

We should reject movement to the player's current location, or to anywhere he hasn't been and can't see:

```
Check going by name:
    if the noun is the location, say "You're already in [the
location]." instead;
    if the noun is not adjacent and the noun is unvisited,
say "That noun did not make sense in this context."
instead.
```

The assumption here is that the player does know the names of the rooms adjacent to his current location, even if he hasn't been there yet.

Now for the travel itself. The simplest way to ensure that the usual movement rules will still apply is to convert GO BY NAME into a GO action, and here the best route comes to our aid:

```
Carry out going by name:
    let aim be the best route from the location to the
noun, using doors;
    if aim is not a direction, say "You can't think how to
get there from here." instead;
    say "(heading [aim])[command clarification break]";
    try going aim;
    if the location is not the noun, say "You'll have to stop
here."
```

This will allow the player to travel toward rooms he has already visited even if they are several moves away.

Finally, so that the player can also use the names of doors as commands:

```
Understand "[door]" as entering.
```

And in keeping with the original, we might add to our scenario a rule or two about restrictions on movement, just to test that it's all working right:

```
The player carries a plover egg and a platinum pyramid.
```

The description of the egg is "Plover's eggs, by the way, are quite large." The printed name of the egg is "emerald the size of a plover's egg". Understand "emerald" as the egg. The description of the pyramid is "The platinum pyramid is 8 inches on a side!"

Instead of going to the Plover Room from the Alcove when the player carries something which is not the plover egg:
    say "Something you're carrying won't fit through the tunnel with you. You'd best take inventory and drop something."

Test me with "go to misty cavern / go to dark corridor / go to plover room / go to alcove / go to dark corridor / drop pyramid / go to dark corridor / g / go to alcove / g / go to misty cavern".

The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling all openable doors on the way.

The foregoing example moves the player one location towards his destination, and requires that rooms have been visited before. But suppose we wanted to be a bit more lenient about movement, and let the player make as many steps as necessary per turn. We will also show consideration about doors, using the "Locksmith" extension supplied with Inform. (Now every time the code attempts opening a door, unlocking rules will also be invoked.)

"Safari Guide"

Include Locksmith by Emily Short.

```
The Monkey House is a room. The African Grasslands
Exhibit is north of the Monkey House. The bird door is
north of the African Grasslands Exhibit and south of the
Aviary. The Ostrich Enclosure is west of the Aviary. The
bird door is a door. It is closed, lockable, and locked.
The silver key is a passkey. It unlocks the bird door. The
player carries the silver key.


Understand "go to [any room]" as going by name.
Understand "[any room]" as going by name. Understand
"[door]" as entering.


Going by name is an action applying to one thing.


Check going by name:
    if the noun is the location, say "You're already in [the
location]." instead.


Carry out going by name:
    while the player is not in the noun:
        let heading be the best route from the location to
the noun, using even locked doors;
        if heading is not a direction, say "You can't think
how to get there from here." instead;
        let destination be the room heading from the
location;
        say "(heading [heading])[command clarification
break]";
        try going heading;
        if the player is not in the destination, rule fails.


Test me with "go to aviary / go to ostrich enclosure /
african grasslands".
```

Notice that we continue the movement until one of two things happens: either the player reaches the room that is his destination, or the going attempt doesn't work. In the latter case we stop the action in order to avoid hanging the game up in a loop. This event might occur when the player runs into a locked door, for instance.

Example **Palette**

RB

An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean") are accepted alongside everyday ones ("blue").

There are hundreds of traditional pigments, from lampblack to burnt sienna, so we will confine ourselves to just two:

```
"Palette"

The Atelier is a room. "The floridly untidy loft space
used by a moderately unsuccessful artist (you, that is)."
The canvas, palette and paint brush are here.
Understand "painting" as the canvas.

Colour is a kind of value. The colours are white, red,
blue and green.

The canvas has a colour. The canvas is white. The
printed name of the canvas is "largely [colour] canvas".

Painting is an action applying to one thing and one
colour. Check painting: if the noun is not the canvas,
say "Centuries of tradition suggest that canvas is the
natural home of paint." instead. Carry out painting: now
the colour of the canvas is the colour understood.
Report painting: say "You splash away at the now
[canvas]."

Understand "paint [something] [a colour]" as painting.
```

Understand "calico" as white. Understand "cerulean" or "cerulean blue" as blue.

Test me with "examine canvas / paint canvas red / examine canvas / paint canvas cerulean / examine canvas".

---

309 **Example** **Baritone, Bass** *RB*

Letting the player pick a gender (or perhaps other characteristics) before starting play.

The "reading a command" activity is rather advanced; for the moment, what we need to understand is that we're intervening in commands at the start of play and insisting that the player's first instruction to the game consist of a choice of gender. After that point, the gender will be set and play will proceed as normal.

In order to do the parsing, we define gender as a kind of value, and give several alternate names to each gender.

"Baritone, Bass"

Getting Started is a room.

Gender is a kind of value. The genders are masculine, feminine, and unknown. Understand "male" or "man" or "M" as masculine. Understand "female" or "woman" or "F" as feminine.

A person has a gender. The gender of the player is unknown.

```
When play begins:
    now the command prompt is "Please choose a
gender for your character. >".



After reading a command when the gender of the player
is unknown:
    if the player's command includes "[gender]":
        now the gender of the player is the gender
understood;
        if the gender of the player is unknown:
            say "This story requires a selection of male or
female. [run paragraph on]";
            reject the player's command;
        if the gender of the player is masculine, now the
player is male;
        if the gender of the player is feminine, now the
player is female;
        say "[line break]Thank you. We now begin...";
        now the command prompt is ">";
        move the player to Sandy Beach;
        reject the player's command;
    otherwise:
        say "Sorry, we're not ready to go on yet. [run
paragraph on]";
        reject the player's command.



Sandy Beach is a room.



Instead of examining the player when the player is
female:
    say "Congratulations, you are a girl!"



Instead of examining the player when the player is male:
    say "Congratulations, you are a boy!"
```

If we had a whole series of things to ask the player about, we might define a whole series of kinds of value

```
The vocal ranges are soprano, mezzosoprano,
```

```
contralto...
```

and use a "construction stage" variable to keep track of the current stage of character-construction, as in

```
After reading a command when the current construction
stage is choosing a vocal range:
    ...
```

---

310 ExampleLies *RB*

Commands to allow the player to lie down in three different ways.

To set the scene, and make new actions to provide for two of these ways:

```
"Lies"


The Laundry is a room. "An old Limehouse haunt, the
Chinese laundry used by the down-trodden wives of the
Tong of the Black Scorpion." The vast marble sink is
here. "There is nothing obviously oriental about the vast
marble sink, which is large enough to lie down inside. A
wooden-rack floor, equipped for easy drainage, turns
out also to be equipped for snagging the shoes of
passers-by." The sink is an enterable container, fixed in
place.


Lying down is an action applying to nothing. Report lying
down: say "You lie down for a while in the middle of the
Laundry, wondering about the point of existence, then
get up again."


Lying near is an action applying to one thing. Report
```

```
lying near: say "You lie down next to [the noun] for a
while, mumbling to yourself."


Instead of lying near the sink, say "Lying down close to
the cool butcher's marble slabs of the sink, your
attention is caught by the sight of coolie shoes through a
floor-level grille for ventilation. The game is afoot!"
```

So far, so good. Now for the grammar, where we create two new tokens: one for each of two groups of alternative prepositions.

```
Understand "beneath/under/by/near/beside/alongside/
against" or "next to" or "in front of" as "[beside]".


Understand "on/in/inside" or "on top of" as "[within]".


Understand "lie down" as lying down.


Understand "lie down [within] [something]" as entering.


Understand "lie [beside] [something]" or "lie down
[beside] [something]" as lying near.


Test me with "lie down / lie down on top of the sink / get
out / lie down inside the sink / get out / lie down in front
of the sink".
```

ExampleAspect

Understanding aspect ratios (a unit) in the names of televisions.

RB

Named properties are not the only kind that Inform is able to understand referring to an object. We can also use unit and number properties to distinguish things from one another, as here, where televisions have aspect ratios:

```
"Aspect"

An aspect ratio is a kind of value. 16:9 specifies an aspect ratio.

A television is a kind of device. A television has an aspect ratio. Understand the aspect ratio property as referring to a television. Understand "European standard" as 16:9.

The Office is a room.

The widescreen TV is a television in the Office. The fifties TV is a television in the Office. The widescreen TV is 16:9. The fifties TV is 4:3.

Test me with "examine european standard tv / x 16:9 tv / x 4:3 tv".
```

ExampleHymenaeus

Understanding "flaming torch" and "extinguished torch" to refer to torches when lit and unlit.

RB

```
"Hymenaeus"
```

A torch is kind of thing. Understand the lit property as describing a torch. Understand "lighted" or "flaming" or "burning" as lit. Understand "extinguished" as unlit. A torch is usually lit.

Before printing the name of a lit torch, say "flaming ".
Before printing the name of an unlit torch, say "extinguished ".

The Wedding Procession is a room.

Orpheus is a man in the Wedding Procession. Orpheus carries a torch.
Eurydice is a woman in the Wedding Procession. Eurydice carries a torch.

Rule for writing a paragraph about someone (called target):
    say "[The target] carries [a list of things carried by the target]."

Every turn:
    if a random chance of 1 in 2 succeeds and a torch is lit:
        let target torch be a random lit torch;
        now the target torch is unlit;
        say "Aquilo blows down from the north, extinguishing the torch carried by [the holder of the target torch]."

Instead of examining a lit torch:
    say "It casts a bright glow over [the holder of the noun]."
Instead of examining an unlit torch:

```
    say "[The holder of the noun] is looking at it
disconsolately, obviously worried about the omens."


Test me with "z / z / z / look / x flaming torch / x
extinguished torch".
```

---

313 **ExampleChannel 1** *RB*

Understanding channels (a number) in the names of televisions.

We might want to allow every television to be tuned to a channel (a number property) which the player could refer to, so that

```
WATCH CHANNEL 13
TURN OFF CHANNEL 4
```

would be directed to the appropriate television object, if any television is turned on and tuned to the correct station. We might now write:

```
"Channel"


A television is a kind of device. A television has a
number called the channel. Understand the channel
property as referring to a television. Understand
"channel" as a television.


The Office is a room. The widescreen TV is a television
in the Office. The fifties TV is a television in the Office.


Changing the channel of it to is an action applying to
one thing and one number.
```

Understand "tune [something] to [number]" or "change channel of [something] to [number]" as changing the channel of it to.

Check changing the channel of something to:
    if the noun is not a television, say "[The noun] cannot be tuned to a channel." instead.

Carry out changing the channel of something to:
    now the channel of the noun is the number understood.

Report changing the channel of something to:
    say "You tune [the noun] to channel [number understood]."

Instead of examining a television:
    if the noun is switched off, say "[The noun] is currently turned off." instead;
    let the chosen channel be the channel of the noun;
    if the chosen channel is a current channel listed in the Table of Television Channels:
        choose row with current channel of the chosen channel in the Table of Television Channels;
        say "[output entry][paragraph break]";
    otherwise:
        say "Snow fills the screen of [the noun]."

Table of Television Channels

output	channel
"The screen of [the noun] is completely black."	0
"A gloomy female news anchor describes the latest car bomb in Baghdad: 104 dead today, and no sign of change."	4
"A couple of contestants in spangled scarlet outfits are performing an energetic paso doble."	5

"On-screen, Ichiro is up to bat with one man on second and no outs."

Test me with "change channel of fifties tv to 4 / x channel 4 / switch on fifties / x channel 4 / switch on widescreen / tune fifties tv to 5 / x channel 5 / x fifties tv / x channel 4".

---

314 ExampleTerracottissima *RB*

The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

This easiest way to do this uses the "printing the name of" activity, which will come up in the following chapter:

"Terracottissima"

A flowerpot is a kind of thing. Understand "pot" as a flowerpot.

A flowerpot can be unbroken or broken. After dropping an unbroken flowerpot: say "Crack!"; now the noun is broken. Understand the broken property as describing a flowerpot.

Before printing the name of a broken flowerpot, say "broken ". Before printing the name of an unbroken flowerpot: if a flowerpot is broken, say "unbroken ".

Before printing the plural name of a broken flowerpot,

say "broken ". Before printing the plural name of an unbroken flowerpot: if a flowerpot is broken, say "unbroken ".

The Herb Garden is a room. In the Herb Garden are ten unbroken flowerpots.

Test me with "get three flowerpots / drop all / look".

---

315 ExamplePeers *RB*

The peers of the English realm come in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince - and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

Almost all of this example is the flummery of pomp and circumstance: only the first two paragraphs really do anything.

"Peers" by Elizabeth II R

A title is a kind of value. The titles are Baron, Viscount, Earl, Marquess, Duke and Prince.

A peer is a kind of man. A peer has a title. A peer is usually a Baron. Before printing the name of a peer, say "[title] ". Understand the title property as describing a peer.

The House of Lords is a room. Maltravers, Pollifax, Omnium and St Vincent are peers in the House of Lords. Omnium is a Duke. St Vincent is an Earl.

Ennobling is an action applying to one thing and one title.

Check ennobling:
    if the noun is the player, say "The Sovereign is the fountain of honour, and may not be ennobled." instead;
    if the noun is not a peer, say "Commoners should remain so." instead;
    if the title of the noun is the title understood, say "But that is his title already." instead;
    if the title of the noun is greater than the title understood, say "As he is already of the rank of [title of the noun], any such letters patent are liable to be deemed invalid, following the precedent of the Buckhurst Peerage Case (1876). Best not." instead.

Carry out ennobling:
    now the title of the noun is the title understood.

Report ennobling:
    say "'Whereas Our Parliament for arduous and urgent affairs concerning Us the state and defence of Our United Kingdom and the Church is now met at Our City of Westminster We strictly enjoining Command you upon the faith and allegiance by which you are bound to Us that the weightness of the said affairs and imminent perils considered (waiving all excuses) you be at the said day and place personally present with Us and with the said Prelates Great Men and Peers to treat and give your counsel upon the affairs aforesaid And this as you regard Us and Our honour and the safety and defence of the said Kingdom and Church and dispatch of the said affairs in nowise do you omit Witness Ourself at Westminster the Fifth day of November in the 43rd year of Our Reign,' you say, with unpunctuated serenity. The new [noun] bows stiffly."

Understand "dub [someone] a/an [title]" as ennobling.

```
Test me with "dub st vincent a baron / dub maltravers a
marquess / look / examine marquess".
```

---

316 **ExampleChannel 2** *RB*

Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

Our previous implementation of televisions ("Channel 1") doesn't allow the player to type things like

```
TUNE FIFTIES TELEVISION TO CHANNEL 4
```

nor does it deal with player input like

```
TUNE TO CHANNEL 4 ON FIFTIES TELEVISION
```

or

```
TUNE TO CHANNEL 4
```

where no television is specified. When we are designing commands which involve two elements (here, a television and a channel number), it's usually a good idea to allow the player to specify those elements in either order, as we saw demonstrated briefly in "New commands for old grammar".

We might, therefore, want to add a few refinements: first by defining a "[channel]" token that will accept input of the forms "[number]" and "channel [number]", and second by creating some additional "Understand" lines that will accept variant versions of the player's input.

```
"Channel 2"
```

Section 1 - Televisions in General

A television is a kind of device.

A television has a number called the channel. Understand the channel property as referring to a television. Understand "channel" as a television.

Changing the channel of it to is an action applying to one thing and one number.

Understand "tune [television] to [channel]" or "change channel of [television] to [channel]" as changing the channel of it to.

Understand "tune [something] to [channel]" or "change channel of [something] to [channel]" as changing the channel of it to.

Understand "tune to [channel] on [television]" or "change to [channel] on [television]" as changing the channel of it to (with nouns reversed).

Understand "tune to [channel] on [something]" or "change to [channel] on [something]" as changing the channel of it to (with nouns reversed).

Understand "[number]" or "channel [number]" as "[channel]".

```
Check changing the channel of something to:
    if the noun is not a television, say "[The noun] cannot
be tuned to a channel." instead.


Carry out changing the channel of something to:
    now the channel of the noun is the number
understood.


Report changing the channel of something to:
    say "You tune [the noun] to channel [number
understood]."


Instead of examining a television:
    if the noun is switched off, say "[The noun] is
currently turned off." instead;
    let the chosen channel be the channel of the noun;
    if the chosen channel is a current channel listed in the
Table of Television Channels:
        choose row with current channel of the chosen
channel in the Table of Television Channels;
        say "[output entry][paragraph break]";
    otherwise:
        say "Snow fills the screen of [the noun]."


Table of Television Channels

output channel
"The screen of [the noun] is completely black." 0


Section 2 - The Scenario


The Office is a room.
```

```
The widescreen TV is a television in the Office. The
fifties TV is a television in the Office.
```

And we add the scenario-specific content to our Table of Television Channels; in the case of channel 13, we provide for a changing sequence of events using text variations.

```
Table of Television Channels (continued)

output channel
4 A gloomy female news anchor describes the latest car bomb in
Baghdad: 104 dead today, and no sign of change."
5 A couple of contestants in spangled scarlet outfits are performing
an energetic paso doble."
13 "[one of]On-screen, Ichiro is up to bat with one man on second
and no outs.[or]Ichiro has singled to first and the other man is on
third.[or]The next batter is in the middle of flying out.[or]Everything
looks rosy until the men in black pull off a double-play and retire
the side.[or]The channel has cut to a commercial.[stopping]"
```

```
Test me with "test one / test two".
```

```
Test one with "change channel of fifties tv to 4 / x
channel 4 / switch on fifties / x channel 4 / switch on
widescreen / tune fifties tv to channel 5 / x channel 5 / x
fifties tv / x channel 4".
```

```
Test two with "tune to channel 13 / widescreen / tune
channel 13 to channel 5 / tune channel 5 to channel 3 /
widescreen / x channel 3".
```

ExampleTerracottissima Maxima
Flowerpots with textual names that might change during play.

Inform can also understand text properties:

```
"Terracottissima Maxima"

A flowerpot is a kind of thing. A flowerpot has a text
called pattern. Understand the pattern property as
describing a flowerpot. The printed name of a flowerpot
is usually "[pattern] flowerpot". The printed plural name
of a flowerpot is usually "[pattern] flowerpots".

The Herb Garden is a room. In the Herb Garden is a
flowerpot with pattern "blue willow". In the Herb Garden
is a flowerpot with pattern "striped". In the Herb Garden
is a flowerpot with pattern "striped".

Test me with "x blue willow / get striped / look".
```

This may not seem very much different from having the pattern be a kind of value -- except that texts can, of course, hold almost anything. Further exploration of these possibilities may be found in the chapter on Advanced Text.

ExampleTilt 1

A deck of cards with fully implemented individual cards, which can be separately drawn and discarded, and referred to by name.

We've simulated a deck of cards before, but only as entries in a table. This time we're going to do it more completely, with card objects that can be drawn and discarded, and referred to by name. The tedious way to do this would be to make 52 objects by hand and laboriously write out their names and understand rules.

A more sensible way is to make 52 identical card objects, assign them ranks and suits, and allow Inform to generate and parse their names automatically.

So:

```
"Tilt"

Section 1 - Cards

Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades. Understand "heart" as hearts.
Understand "club" as clubs. Understand "diamond" as
diamonds. Understand "spade" as spades. [Providing
the singular forms means that Inform will also
understand >EXAMINE SPADE, >DISCARD CLUB, and
so on.]

A card is a kind of thing. A card has a suit. A card has a
number called rank. Understand the suit property as
describing a card. Understand the rank property as
describing a card.

52 cards are in the card repository.
```

Now, we're going to describe the higher numbers as face cards, so it helps to write a new "to say" phrase, just as we did in Jokers Wild. (A subsequent version of this example shows how to print card values with red and black symbols representing the different suits; see "Tilt 3".)

```
To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".

Rule for printing the name of a card (called target):
    say "[rank of the target as a card value] of [suit of the
target]"
```

```
Table of Value Names

topic                term
"ace/A/one"          "1"
"deuce/two"          "2"
"three"              "3"
"four"               "4"
"five"               "5"
"six"                "6"
"seven"              "7"
"eight"              "8"
"nine"               "9"
"ten"                "10"
"jack/knave/J"       "11"
"queen/Q"            "12"
"king/K"             "13"
```

This is enough already to let inform understand things like "ten clubs", but we want to add a couple of refinements. For one thing, we'd like to accept "of" when it appears in phrases such as "ten of clubs" (but not generically otherwise); for another, we'd like the player to be able to use various names for ranks. To this end, we need to borrow from the Activities chapter and modify the player's command before attempting to understand it:

```
After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
            cut the matched text;
    repeat through the Table of Value Names:
        while the player's command includes topic entry:
            replace the matched text with value entry.
    [This allows Inform to understand "ace", "deuce",
"king", etc., as numerical ranks.]
```

It may be a bit confusing that the Table of Value Names has both a topic column and a term column, to all appearances essentially identical. But items in the topic column can be matched against the player's input, whereas items in other kinds of text column can be printed out; the two kinds of text are not treated identically by Inform, so we need to have both. Notice that the topic column contains entries like "jack/knave," which will match either "jack" or "knave" in the player's input.

Now to set up the deck at the outset. With some intelligent looping, we

avoid having to declare every combination of suit and number individually:

```
When play begins:
    reconstitute deck.


To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;
            now item is in the deck of cards;
        now current suit is the suit after the current suit.
```

And now we need a simple setting and some actions to manipulate the deck with:

```
Section 2 - The Deck and the Discard Pile


The Empty Room is a room. "Nothing to see here."


The deck of cards is in the Empty Room. It is a closed
unopenable container. The description is "A standard
poker deck."


The discard pile is a closed unopenable container. The
description is "Cards in this game are discarded face-
down, so the discard pile is not very interesting to see.
All you can observe is that it currently contains [if the
number of cards which are in the discard pile is less
than ten][the number of cards which are in the discard
pile in words][otherwise]about [the rounded number of
cards which are in the discard pile in words][end if]
card[s]."
```

```
To decide what number is the rounded number of
(described set - a description of objects):
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.
```

The above phrase rounds a number to the nearest five, so that the player is not autistically able to count a large number of cards in the discard pile at a single glance.

This next bit is an optional borrowing from the Activities chapter: we want to prevent Inform printing things like "You can see a discard pile (closed) here.", since we don't want the player to think of the piles as containers, even though Inform thinks of them in those terms.

```
Rule for printing room description details of something:
do nothing instead.
```

Finally, we want the player to use "draw" and "discard" to manipulate his hand of cards:

```
Section 3 - Drawing and Discarding Actions

Understand the commands "take" and "carry" and "hold"
and "get" and "drop" and "throw" and "discard" as
something new.

Understand "take [text]" or "get [text]" or "drop [text]" as
a mistake ("Here, you only draw and discard. Nothing
else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as
drawing. Drawing is an action applying to nothing. The
drawing action has an object called the card drawn.
```

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

Understand "discard [card]" as discarding. Discarding is an action applying to one thing.

Check discarding:
    if the player does not carry the noun, say "You can only discard cards from your own hand." instead.

Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile to the location.

```
Report discarding:
    say "You toss [the noun] nonchalantly onto the
discard pile."

Seeding is an action out of world. Understand "seed" as
seeding. Carry out seeding: seed the random-number
generator with 5681.

Test me with "seed / draw / g / g / g / g / i / discard
seven of spades / draw / discard six / draw / i / discard
hearts / discard six of diamonds card / draw / draw / i /
discard spades card / draw / discard king card".
```

ExampleCinco

RB

A taco shell that can be referred to (when it contains things) in terms of its contents.

It's fairly common that we want to be able to refer to a container in terms of what it has in it: a bottle of wine, a salt shaker, a chicken sandwich. The player is free to remove the contents again, and the object will go back to using its usual name:

```
"Cinco"

Cinco de Mayo Fundraiser is a room.

The taco shell is an edible thing in the Fundraiser. It is a
portable container. It has carrying capacity 1.
```

```
Understand "[something related by containment] taco"
as the taco.


Rule for printing the name of the taco shell while not
inserting or removing:
    if the taco contains something (called filling), say
"[filling] taco";
    otherwise say "taco shell";
    omit contents in listing.


The player carries shredded beef. It is edible.


The taking action has an object called source (matched
as "from").


Setting action variables for taking:
    now source is the holder of the noun.


Report taking something from the taco shell:
    say "You gingerly pick [the noun] out of the taco
shell." instead.


Test me with "x taco / put shredded beef in taco / get
taco / i / x shredded beef taco / get shredded beef / x
shredded beef taco".
```

---

320 **ExamplePuncak Jaya** *RB*

When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

```
By default, when something is not present, Inform does not allow a player
```

to refer to it. But there are times when we might like to acknowledge that the thing mentioned in a command does exist somewhere in the game; it just happens not to be on hand right now.

One way to do this is to make an object that appears everywhere and responds to the name of its owner only when the owner itself is not in view.

```
"Puncak Jaya"


A ghost is a kind of person. A man-ghost is a kind of
ghost. A man-ghost is always male. A woman-ghost is a
kind of ghost. A woman-ghost is always female.
```

We make the ghost a person rather than some other kind of thing so that it will be able to respond to commands such as KISS BOB or (even trickier) BOB, JUMP: if Inform did not recognize the ghost as an animate creature, it would not accept such input.

```
Representation relates one ghost to one person. The
verb to represent means the representation relation.


One man-ghost represents every man. One woman-
ghost represents every woman.
```

This is, technically, an assembly -- except instead of saying that every device has a button part, or that there are three daffodils in every garden room, the assembly is based on a non-physical relation that we just designed.

Based on the "representation" relation, we now devise a conditional relation that applies only when the represented thing is not itself in view:

```
Indication relates a ghost (called X) to a person (called
Y) when X represents Y and Y is not visible.
```

```
Understand "[something related by indication]" as a
ghost.

When play begins:
    now every ghost is in the concept-repository.

Instead of doing something to a ghost:
    say "You seem to have left [a random person which is
represented by the noun] behind."

Instead of doing something when the second noun is a
ghost:
    say "You seem to have left [a random person which is
represented by the second noun] behind."

The concept-repository is an open unopenable
transparent container. It is part of the air. The air is a
backdrop. It is everywhere.

Base of Puncak Jaya is a room. Temple, Kippax, and
Huizenga are men in Base. Peak of Puncak Jaya is
above Base of Puncak Jaya.

Test me with "x kippax / up / x kippax / kiss kippax /
kippax, hello".
```

Further complications of this example might require that the player meet a character before being able to refer to him or her.

---

321 ExampleWhither? *RB*

A door whose description says where it leads; and which automatically understands references such as "the west door" and "the east door" depending on which direction it

leads from the location.

Here we expand on the simple examples When? and Whence?; this time we want the player to be able to refer to doors by their directions, as in "the west door" when the door in question does in fact lead west.

```
"Whither?"

The temporal vortex is an open door. It is west of
Yesterday and east of Today.

The initial appearance of a door is usually "Nearby [an
item described] leads [if the other side of the item
described is visited][direction of the item described from
the location] to [the other side][otherwise][direction of
the item described from the location][end if]."

Direction-relevance relates a door (called X) to a
direction (called Y) when the direction of X from the
location is Y. The verb to be directionally-relevant to
means the direction-relevance relation.

Understand "[something related by direction-relevance]
door" as a door.
```

As an added touch, we respond also to the case where the player postulates a door in some direction when there is no such thing at the moment:

```
Rule for printing a parser error when the player's
command includes "[non-door direction] door":
    say "There is no door in that direction." instead.
```

```
Definition: a direction (called direction D) is non-door:
    let the target be the room-or-door direction D from the
location;
    if the target is a door:
        no;
    yes;
```

```
Test me with "examine west door / x east door / w / x w
door / x e door / tie me to the west door / tie the west
door to me / push the west door east / push the east
door west".
```

---

322 **ExampleClaims Adjustment** *RB*

An instant camera that spits out photographs of anything the player chooses to take a picture of.

We start by creating a camera and a photograph object. As usual when we want to have a kind of object that can be dispensed in bulk, we start off with a bunch of identical instances of the object out of play (in this case, kept in an out-of-play container called "film roll"); we can then move them into play and give them characteristics when they're needed.

Each photograph can depict exactly one thing -- we're assuming that the player is not a landscape photographer here -- so we create a relation to indicate what is shown by each photograph. We'll then use that relation to determine how photographs are described, named, and parsed:

```
"Claims Adjustment"
```

```
A photograph is a kind of thing. 36 photographs are in
the film roll.
```

```
Appearance relates one thing to various photographs.
The verb to be shown by means the appearance
relation.
```

```
The description of a photograph is usually "It shows [a
random thing which is shown by the item described]."

Understand "of [something related by reversed
appearance]" as a photograph.
```

This allows the player to refer to any photograph by its subject: useful if we have a large number of them.

Now we create an action to let the player use the camera and generate these photograph objects:

```
The player carries a cheap instant camera.

Understand "photograph [something] with [camera]" as
photographing. Understand "photograph [something]
with [something preferably held]" as photographing.
Photographing is an action applying to one visible thing
and one carried thing, requiring light.

The photographing action has an object called the
selected film.

Setting action variables for photographing:
    let N be a random photograph in the film roll;
    now the selected film is N.

Check photographing:
    if the second noun is not the camera, say "You need
a camera for that purpose." instead.

Check photographing:
```

```
    if the noun is the camera, say "Sadly impossible."
instead.


Check photographing:
    if the selected film is nothing, say "You're out of film."
instead.


Carry out photographing:
    now the noun is shown by the selected film;
    move the selected film to the player.


Report photographing:
    say "Your camera instantly spits out [a selected film]."
```

Now we use two activities from the Activities chapter to describe the photographs to the player more elegantly:

```
After printing the name of a photograph (called target):
    say " of [a random thing which is shown by the
target]".


After printing the plural name of a photograph (called
target):
    let N be the holder of the target;
    say " of [a list of things which are shown by
photographs which are held by N]";
    if the number of things which are shown by
photographs which are held by N is greater than one,
say " (variously)".
```

And finally we provide a brief scenario to give the player something to take pictures of:

```
The Treasure Room is a room. "Despite the fancy
name, this is no more than a closet -- albeit a closet with
its own special circuit on the house alarm."
```

The Treasure Room contains a small Degas, a Ming vase, and a collection of South African krugerrands. The player is carrying insurance forms, a first-class stamp, and a security envelope.

The description of the forms is "Completely filled out in black ink in block letters: now all you need to do is attach photographic evidence of the objects you wish to insure."

Test me with "photograph degas / i / photograph degas / i / x photograph of degas / photograph me / x photograph of me / i / photograph vase / photograph camera / photograph collection / g / i / test more".

Test more with "x photograph of collection / x photograph of krugerrands / x photograph of collection of south african krugerrands / photograph photograph of degas / x photograph of photograph of degas".

---

323 **Example** **Quiz Show** *RB*

In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

"Quiz Show" by Mike Tarbert

Use scoring.

```
Answer mode is a truth state that varies.
Current state is a text that varies.
```

```
Guessing is an action applying to one topic.
Understand "[text]" as guessing when answer mode is
true.
```

Because of the "...when" part of this line, random text is only treated as an answer when a question is being asked.

```
Check guessing (this is the default wrong answer rule):
    if the topic understood is not a topic listed in the Table
of Dates of Statehood:
        say "Wrong!";
    now answer mode is false.
```

```
Carry out guessing a topic listed in the Table of Dates of
Statehood:
    if state entry is the current state:
        say "Correct! ([comment entry], to be exact!)";
        increase the score by one;
    otherwise:
        say "Wrong!";
    now answer mode is false.
```

This next rule allows a player to do something other than answer the question, but then makes him wait for another question before answering.

```
Before doing anything other than guessing:
    if answer mode is true:
        say "(ignoring the question)[line break]";
    now answer mode is false.
```

```
Section 2 - Scenario
```

```
The Lab is a room. Sam is a man in the lab.


Every turn when the player is in the lab:
    if a random chance of 3 in 5 succeeds:
        choose a random row in the Table of Dates of
Statehood;
        say "Sam asks you, 'In what year was [state entry]
admitted into the Union?'";
        now current state is state entry;
        now answer mode is true.


Table of Dates of Statehood

State                Entry
"Maryland"           1788
"Delaware"           1787
"Pennsylvania"       1787
"New Jersey"         1787
"Georgia"            1788
"Connecticut"        1788
"Massachusetts"      1788
"Maine"              1820
"South Carolina"     1788
"New Hampshire"      1788
"Virginia"           1788
"New York"           1788
"North Carolina"     1789
"Rhode Island"       1790
"Vermont"            1791
"Kentucky"           1792
"Tennessee"          1796
"Ohio"               1803
"Louisiana"          1812
"Indiana"            1816
"Mississippi"        1817
"Illinois"           1818
"Alabama"            1819
"Missouri"           1821
"Arkansas"           1836
"Michigan"           1837
"Florida"            1845
"Texas"              1845
"Iowa"               1846
"Wisconsin"          1848
"California"         1850
"Minnesota"          1858
"Oregon"             1859
"Kansas"             1861
"West Virginia"      1863
"Nevada"             1864
"Nebraska"           1867
"Colorado"           1876
"North Dakota"       1889
"South Dakota"       1889
"Montana"            1889
"Washington"         1889
"Idaho"              1890
"Wyoming"            1890
"Utah"               1896
"Oklahoma"           1907
"New Mexico"         1912
"Arizona"            1912
"Alaska"             1959
"Hawaii"             1959

Test me with "1845 / z / z / 1787 / 1792 / z / 1845 / g".
```

Note that the situation will become a little more complicated if we have two or more identical topics in our trivia list; in that case, we would need to loop through the Table of Dates of Statehood explicitly, and only mark the player wrong if none of the lines were found to match. (See the chapter on Tables for many more ways to manipulate table behavior.)

ExampleBibliophilia *RB*

A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

Suppose we want a bookshelf with a very large number of books on it. They aren't to be taken or carried around in the game, but they should be mentioned, and the player should be allowed to look them up by name. Furthermore, the player's attempts to examine something unrecognized should be understood as an attempt to look up a title -- but only when the player is in the presence of the books. The rest of the time such requests

should be rejected in the usual way.

"Bibliophilia"

The Graduate Lounge is a room. "Shabby sofas; plastic cups remaining from the afternoon's pre-lecture espresso; a collection of Xena and Hercules figurines posed for ironic effect. It's somewhat depressing at this hour, when everyone has gone home."

The Classics Reading Room is south of the Lounge. "Not as large a collection as the one in the Library, but it contains copies of everything really essential for reference."

Understand "examine [text]" as examining as a book when the player is in the Reading Room. Understand "look up [text]" as examining as a book when the player is in the Reading Room.

Examining as a book is an action applying to one topic.

Carry out examining as a book:
    say "You can't find any such text."

Instead of examining as a book a topic listed in the Table of Book Titles:
    say "[description entry][paragraph break]"

Table of Book Titles

title description

"Reading Greek Death" "reading/greek/death" "A dense orange paperback treatise on the development of Greek eschatology."
"TAPA 134:2" "tapa/134-2" "Transactions of the American Philological Association from 2004."
"Oxford Classical Dictionary" "OCD/dictionary/classical/oxford" "A hefty reference with short articles on everything from classical Greek meter to ancient cosmetics."
"Collected Dialogues of Plato" "plato/dialogues/hamilton/cairns" "All the Platonic dialogues -- some, admittedly, in rather tired translations -- but still a useful single volume, ed. Edith Hamilton and Huntington Cairns."
"Adobe Illustrator CS User Guide" "adobe/illustrator/cs/user/guide" "Hello, how did this get here? A suspiciously familiar name is scribbled inside the front cover."

Some books are scenery in the Reading Room.
Understand "copies" or "book" or "shelf" or "shelves" as the books. Instead of examining the books:
    choose a random row in the Table of Book Titles;
    say "You scan the shelves and notice, among others, a volume entitled [italic type][title entry][roman type]."

Test me with "south / examine ocd / examine books / examine books / examine plato / n / x hercules / s / x hercules".

Now if we type >X HERCULES in the Lounge, we will get

>x hercules
You can't see any such thing.

thanks to our somewhat slovenly implementation of the Lounge scenery; but in the Reading Room,

>x hercules
You can't find any such text.

In practice we might also want to extend our coverage somewhat to handle a case where the player tried to take books from the bookshelf: currently that would not be understood.

Example Pot of Petunias

Responding sensibly to a pot of petunias falling from the sky.

Suppose we have an object that makes a dramatic entrance on the scene, like so:

```
"Pot of Petunias"

Wide Open Field is a room. "A big field under a big sky.
The clouds are puffy, the trees are handsome."

Some clouds and some trees are scenery in Wide Open
Field. The description of the clouds is "That one looks
like Yoda's head." The description of the trees is
"You've never been much good at botany, so it's
anyone's guess what kind they are."

A rock is in Wide Open Field. The description of the rock
is "It looks like it's been here from the dawn of time."

The broken flower pot is a thing. The description of the
broken flower pot is "It contains the remains of some
abused petunias."

At 9:01 am:
    move the broken flower pot to the location;
    say "Quite unexpectedly, a flower pot falls from the
sky and breaks open on the ground. Good thing you
weren't standing six inches to the left.";
    set pronouns from the broken flower pot.

Test me with "x it / x it / x it".
```

If we leave out the "set pronouns..." line here, we'll wind up with the following very unsatisfactory end to our test transcript:

```
Quite unexpectedly, a flower pot falls from the sky and
breaks open on the ground. Good thing you weren't
standing six inches to the left.


>[3] x it
It looks like it's been here from the dawn of time.
```

Example **Masochism Deli** *RB*

Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

Here the player has several potatoes; we would like to make him more likely to drop the hot one, and more likely to pick up the cold one, all else being equal. At the same time, we want to phrase our rules so that they don't make the player try to take something he's already holding, or drop something he isn't.

So:

```
"Masochism Deli"


The Masochism Deli is a room. "Recent restructurings
of corporate policy restrict the 'lunch hour' to exactly
thirty-two minutes, which means that no one has time to
go out. Instead, you and your coworkers eat here, in the
company's very own themed lunch room."


The plural of potato is potatoes. A potato is a kind of
thing. A potato is edible.
```

Temperature is a kind of value. The temperatures are hot and cold. A potato has a temperature. A potato is usually cold.

Understand the temperature property as describing a potato. Before printing the name of a potato (called subject): say "[temperature of subject] ". Before printing the plural name of a potato (called subject): say "[temperature of subject] ".

Does the player mean dropping a hot potato which is carried by the player: it is very likely.

Does the player mean taking a cold potato which is not carried by the player: it is very likely.

The player carries three potatoes.

After dropping a hot potato:
    say "The guy from Cube B sneers at your lack of potato-holding stamina."

When play begins: now a random potato is hot.

Test me with "inventory / drop potato / g / g / get potato / g / i / get potato".

Example **The Best Till Last**
Reordering multiple objects for dramatic effect.

If a single command asks to do many things, some dull and some exciting, we may want to save the good ones for the end.

"The Best Till Last"

The Funky Ignition Lounge is a room. "This is where all evenings end." The stick of gelignite, the solid magnesium footstool, the vetiver candle, and the vodka bottle are here.

The burn description of the vetiver candle is "It burns right down, expensively but gothically."

The player carries an inexpensive firework. The description of the firework is "It is a cardboard tube with red and green stripes along the outside, and a fuse sticking out of the end." The burn description of the firework is "It ignites gloriously! You take a few hasty steps back in order to avoid burning yourself, and not a moment too soon. Red and green sparks fly out of the tube, and there's a whistling noise punctuated by several loud cracks."

The player carries a lighter. The description of the lighter is "You don't smoke, but you like to have access to flame now and then anyway."

Burning it with is an action applying to one thing and one carried thing.

Understand "burn [things] with [something preferably held]" as burning it with.

```
The block burning rule is not listed in any rulebook.


A thing has some text called the burn description.


Check burning something:
    if the player carries the lighter:
        try burning the noun with the lighter;
    else:
        try burning the noun with the noun.


Check burning something with something when the
second noun is not the lighter:
    say "Your trusty lighter is the best flame source
available to you." instead.


Check burning something with something:
    if the burn description of the noun is "":
        say "Best not." instead.


Carry out burning something with something:
    remove the noun from play.


Report burning something with something:
    say "[burn description of the noun][line break]".


A multiple action processing rule when the action name
part of the current action is the burning it with action
(this is the orderly burn rule):
    let L be the multiple object list;
    let dull list be a list of objects;
    let fun list be a list of objects;
    repeat with item running through L:
```

```
        if the burn description of the item is "":
            add item to dull list;
        else:
            add item to fun list;
    let F be the dull list;
    add fun list to F;
    alter the multiple object list to F.


Test me with "burn all with lighter".
```

ExampleWestern Art History 305 *RB*

Allowing EXAMINE to see multiple objects with a single command.

In a gallery, there are many individual things to look at, but you can also get a general impression by just examining them as a collection.

First, we'll make a kind for the paintings exhibited in the gallery, and then we'll also make a special object to represent all of them as a mass:

```
"Western Art History 305"


A painting is a kind of thing. A painting is usually fixed in
place. Understand "painting" as a painting. Understand
"paintings" as the plural of painting.


The painting-collective is a thing. The printed name of
the painting-collective is "paintings". The description of
the painting-collective is "There's [a list of visible
paintings]."
```

We could if we wanted tweak the description to be different in style in different rooms of the gallery, but this will do for now. Next we need to make it possible to type something like EXAMINE PAINTINGS, which normally wouldn't work because the Standard Rules don't tell Inform to recognise multiple objects with the EXAMINE command (unlike, say,

DROP or TAKE). This is easy:

```
Understand "examine [things]" as examining.
```

Now to make use of the special object. If the player types EXAMINE PAINTINGS, the multiple object list will become a list of the visible paintings. The following rule looks at this list: if it contains more than one painting, it replaces them with the painting-collective instead. Now there's only one examining action, so we get a reply like "There's an abstract painting, a pointilist painting and a French academic painting." instead of a list of descriptions of each in turn.

```
A multiple action processing rule when the current
action is examining (this is the examine kinds rule):
    let L be the multiple object list;
    let F be L;
    let the painting count be 0;
    repeat with item running through L:
        if the item is a painting:
            increment the painting count;
            remove the item from F;
    if the painting count is greater than one:
        add the painting-collective to F;
        alter the multiple object list to F.
```

And now some art to try this out on:

```
Gallery is a room. "Various paintings hang on the walls
of this gallery, awaiting critical attention. A side chamber
to the north contains smaller works."
```

```
The abstract painting, the pointilist painting, and the
French academic painting are paintings in the Gallery.
```

```
North of the Gallery is the Side Chamber. A handsome
miniature is a painting in the Side Chamber. The
description of the handsome miniature is "The miniature
depicts a uniformed soldier of the late 18th century, with
braid on his shoulders and a curl in his beard."
```

The player carries a small notebook. The description of the notebook is "It contains the notes you've taken so far towards a paper for Western Art History 305. So far you're still feeling a bit uninspired."

Test me with "x paintings / x all / n / x paintings / x all".

---

329 **ExampleQuery** *RB*

Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

First, we create a single "[query]" token so that we can capture all instances of such sentences in a single line:

"Query"

Blank Room is a room.

Understand "who" or "what" or "when" or "where" or "why" or "how" or "who's" or "what's" or "when's" or "where's" or "why's" or "how's" as "[query]".

Understand "[query] [text]" as a mistake ("[story title] understands commands, such as '[command prompt]examine [a random thing that can be seen by the player]', but not questions. For more instructions, type HELP.").

Test me with "who am I? / who are you? / where is this

place?".

Now the game will respond to all questions novice players might type with this reminder to look for help information.

---

330 **Example** **The Gorge at George** *RB*

If the player tries to TALK TO a character, suggest alternative modes of conversation.

"Gorge at George"

The Dusty Lot is a room. "A few miles up the road from the concert venue, but at least it's cheap to park here."

The motorcyclist is a man in the Dusty Lot. "A man clad in [a list of things worn by the motorcyclist] leans against his Harley and watches you without saying anything." The Harley is scenery in the Lot. The motorcyclist wears a black leather jacket and shades. Understand "man" or "guy" as the motorcyclist.

Understand "talk to [someone]" as a mistake ("To start a conversation, try to ASK [the noun] ABOUT something or TELL [the noun] ABOUT something.").

Instead of asking the motorcyclist about something:
    say "He smirks cryptically."

Instead of telling the motorcyclist about something:
    say "This does not seem to interest him much."

Test me with "talk to motorcyclist / ask motorcyclist about himself / tell motorcyclist about me".

---

331 **Example** **Hot Glass Looks Like Cold Glass** *RB*

Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

Suppose we have a situation where the player is allowed to talk about the heat of an object only if he's properly equipped to detect it.

"Hot Glass Looks Like Cold Glass"

Use scoring.

Heat is a kind of value. The heats are hot, warm, room temperature, and cold. A thing has a heat.

Understand the heat property as referring to a thing when the player wears the infrared goggles.

The Test Kitchen is a room. "Your own personal lab, ready for scrupulously scientific recipe research. You hope. The previous three runs of this did not go well." The pair of infrared goggles is carried by the player. The description is "A pair of head-mounted IR goggles which look very foolish when worn." The goggles are wearable.

A glass dish is a kind of container. A glass dish is transparent. Three room temperature glass dishes are on the counter. Two hot glass dishes are on the counter.

```
Two cold glass dishes are on the counter. The counter
is scenery in the Test Kitchen.


Instead of doing something other than examining to a
hot glass dish:
    say "Ow! Crikey! You swear, and Claudia makes a
sympathetic hiss. 'You're going to have a mark from that
for sure,' she comments.";
    decrease the score by 2.


Instead of doing something when a hot glass dish is the
second noun:
    say "You brush [the second noun], and wince, but
manage to conceal that from Claudia.";
    decrement the score.


Before printing the name of a glass dish when the player
wears the goggles: say "[heat] "


Before printing the plural name of a glass dish when the
player wears the goggles: say "[heat] "
```

So far, so good. Now, what if the player tries to GET HOT DISH when the goggles are off? "You can't see any such thing." doesn't seem like quite the right response: he can see such a thing. He just doesn't know which it is.

We could go on to write a mistake rule that would scold the player for trying "get [heat] [text]" when not wearing the goggles. The problem is that this would not cover any other phrasing of the command, nor would it account for all the many other things the player might try to do with an object specified by heat.

What we really want is to catch all instances of the player using the property name when not allowed to do so; and for this purpose we can borrow a trick from the chapter on Activities:

```
After reading a command:
    if the player wears the goggles, make no decision;
```

```
    if the player's command includes "[heat]":
        say "Without the IR goggles on, you cannot tell hot
things from cold at sight.";
        rule succeeds.
```

```
Claudia is a woman in the Test Kitchen. "Your assistant
Claudia stands by with [a list of things carried by
Claudia]." The description of Claudia is "Infinitely patient
and a very good stenographer. She is studiously
avoiding giving you any sort of look that might be
construed as mocking." Claudia carries a notepad, a
brined chicken breast, a blowtorch, and a cup of heavy
cream.
```

```
Instead of asking Claudia for something which is carried
by Claudia:
    move the second noun to the player;
    say "'Check, [second noun],' repeats Claudia, in the
tone of one who has seen too many medical dramas.
She does hand it over, though."
```

```
Instead of asking Claudia for something:
    say "She clears her throat faintly and glances at [the
second noun], as though to say that it's not hers to
give."
```

```
Test me with "get all / drop all / look / wear goggles /
look / ask claudia for cream / put cream in hot dish / put
cream in cold dish / remove goggles / get hot dish".
```

---

332 Example Some Assembly Required *RB*

Building different styles of shirt from component sleeves and collars.

We now have the mechanisms in place to do some fairly sophisticated renaming of objects. For instance:

"Some Assembly Required"

Garment type is a kind of value. The garment types are vest, t-shirt, polo shirt, mandarin blouse, button-down, shell, experiment.

Every turn:
    assign identities.

When play begins: assign identities.

To assign identities:
    repeat with item running through torsos:
        reassess item.

To reassess (item - a torso):
    if the number of things which are part of the item is 0:
        now garment type of the item is vest;
        rule succeeds;
    if exactly two short sleeves are part of the item:
        if a collar is part of the item,
            now garment type of the item is polo shirt;
        otherwise now garment type of the item is t-shirt;
        rule succeeds;
    if exactly two long sleeves are part of the item:
        if a collar is part of the item,
            now garment type of the item is button-down;
        otherwise now garment type of the item is mandarin blouse;
        rule succeeds;
    if a collar is part of the item and the number of sleeves which are part of the item is 0, now garment type of the item is shell;
    otherwise now garment type of the item is experiment.

Before cutting something which is worn by the player:
    try taking off the noun.

Instead of cutting something when something is part of the noun:
    say "You cut up [the noun], snipping off [a list of things which are part of the noun].";
    now every thing which is part of the noun is in the holder of the noun.

Instead of cutting something which is part of something:
    say "You carefully snip [the noun] free.";
    now the player carries the noun.

Rule for printing the name of a torso: say "[garment type]".

A torso is a kind of thing. A torso is always wearable. Understand "shirt" or "blouse" as a torso. A torso has a garment type. Understand the garment type property as describing a torso. A sleeve is a kind of thing. A short sleeve is a kind of sleeve. A long sleeve is a kind of sleeve. A collar is a kind of thing.

Understand "sew [something] to [something]" as affixing it to. Affixing it to is an action applying to two things. Carry out affixing something to something: now the noun is part of the second noun. Report affixing something to something: assign identities; say "You sew [the noun] on, creating [a second noun]." Understand the command "stitch" as "sew".

Instead of affixing something to something when the second noun is worn: say "You're wearing [the second

```
noun]!"


Instead of affixing a torso to something:
    if the second noun is a torso, say "Couture for
Siamese twins is a daring field, but a bit of a niche
market.";
    otherwise try affixing the second noun to the noun.


Instead of affixing a sleeve to something when at least
two sleeves are part of the second noun: say "[The
second noun] already sports [a list of sleeves that are
part of the second noun]."


Instead of affixing a collar to something when a collar is
part of the second noun: say "[The second noun]
already sports [a list of collars that are part of the
second noun]."


Instead of examining something when something is part
of the noun: say "Stitched to [the noun] [is-are a list of
things which are part of the noun]."
```

Here is where the issue of precedence arises. We want to encourage Inform to select a cuttable object that is part of something else, rather than one of the spares:

```
Definition: a thing is removable if it is part of something.
Understand "cut [removable thing]" as cutting.


The Boutique is a room. "Still festively strewn with the
confetti and streamers of the Grand Opening party, and
still almost totally customer-free."


The player carries a torso. The player carries three short
```

sleeves. The player carries two long sleeves. The player carries two collars.

Test me with "sew collar to shirt / i / sew short sleeve to shirt / g / i / x polo shirt / cut collar / i / cut shirt / sew long sleeve to shirt / i / sew long sleeve to shirt / i / sew collar to shirt / g / i / wear button-down".

---

333 **Example** **Lakeside Living** *RB*

Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

Much of what follows is identical to "Lemonade" earlier; the new material begins at Part 2.

"Lakeside Living"

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The current volume of a fluid container is usually 0.0 fl oz.

Liquid is a kind of value. The liquids are water, absinthe,

```
and iced tea. A fluid container has a liquid.


Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun]
at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [liquid of the noun]."


To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3
and tenths part of amount is less than 7:
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in
words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase
amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid
ounce[s]".


Before printing the name of a fluid container (called the
target) while not drinking or pouring:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.


After printing the name of a fluid container (called the
target) while not examining or pouring:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.


Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
```

```
accept anything but liquids."


Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.


Understand "drink from [fluid container]" as drinking.


Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within."
instead;
    otherwise:
        decrease the current volume of the noun by 0.2 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "You take a sip of [the liquid of the noun][if the
noun is empty], leaving [the noun] empty[end if]."


Part 2 - Filling


Understand the command "fill" as something new.
```

Here we want Inform to prefer full liquid sources to other containers when it chooses an end to a player's unfinished or ambiguous command. And so:

```
Understand "fill [fluid container] with/from [full liquid
source]" as filling it with. Understand "fill [fluid container]
with/from [fluid container]" as filling it with.
```

Both grammar lines point to the same ultimate outcome; the purpose of specifying both is to tell Inform to check thoroughly for full liquid sources before falling back on other fluid containers when making its decisions.

Understand "fill [something] with/from [something]" as filling it with.

Filling it with is an action applying to two things. Carry out filling it with: try pouring the second noun into the noun instead.

Understand "pour [fluid container] in/into/on/onto [fluid container]" as pouring it into. Understand "empty [fluid container] into [fluid container]" as pouring it into.

Understand "pour [something] in/into/on/onto [something]" as pouring it into. Understand "empty [something] into [something]" as pouring it into.

Pouring it into is an action applying to two things.

Check pouring it into:
    if the noun is not a fluid container, say "You can't pour [the noun]." instead;
    if the second noun is not a fluid container, say "You can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second noun:
        if the second noun is empty, now the liquid of the second noun is the liquid of the noun;
        otherwise say "Mixing [the liquid of the noun] with [the liquid of the second noun] would give unsavory results." instead;
    if the noun is empty, say "No more [liquid of the noun] remains in [the noun]." instead;
    if the second noun is full, say "[The second noun] cannot contain any more than it already holds." instead.

```
Carry out pouring it into:
    let available capacity be the fluid capacity of the
second noun minus the current volume of the second
noun;
    if the available capacity is greater than the current
volume of the noun, now the available capacity is the
current volume of the noun;
    increase the current volume of the second noun by
available capacity;
    decrease the current volume of the noun by available
capacity.
```

```
Report pouring it into:
    say "[if the noun is empty][The noun] is now empty;
[otherwise][The noun] now contains [current volume of
the noun in rough terms] of [liquid of the noun]; [end if]";
    say "[the second noun] contains [current volume of
the second noun in rough terms] of [liquid of the second
noun][if the second noun is full], and is now full[end if]."
```

```
Understand the liquid property as describing a fluid
container. Understand "of" as a fluid container.
```

And now we add our liquid source kind, which will represent lakes, absinthe fountains, and any other infinite supplies of liquid we might need. Note that 3276.7 is the largest possible number of fluid ounces available to us.

```
A liquid source is a kind of fluid container. A liquid
source has a liquid. A liquid source is usually scenery.
The fluid capacity of a liquid source is usually 3276.7 fl
oz. The current volume of a liquid source is usually
3276.7 fl oz. Instead of examining a liquid source: say
"[The noun] is full of [liquid of the noun]."
```

```
Carry out pouring a liquid source into something: now
the current volume of the noun is 3276.7 fl oz.
```

We want filling things from liquid sources to work the same way as usual,

with the distinction that a) the liquid source never depletes in quantity (hence the carry-out rule resetting its fullness); and b) we should report the results a bit differently as well:

```
After pouring a liquid source into a fluid container:
    say "You fill [the second noun] up with [liquid of the
noun] from [the noun]."
```

On the other hand, pouring liquids into a liquid source needs to work completely differently from pouring liquids into anything else. Let's say we're going to allow any liquid at all to be dumped into rivers and streams (environmental protections evidently are not very well-enforced in this scenario):

```
Instead of pouring a fluid container into a liquid source:
    if the noun is empty, say "[The noun] is already
empty." instead;
    now the current volume of the noun is 0.0 fl oz;
    say "You dump out [the noun] into [the second
noun]."
```

A couple of minor refinements:

```
Swimming is an action applying to nothing. Understand
"swim" or "dive" as swimming.
```

```
Instead of swimming in the presence of a liquid source:
    say "You don't feel like a dip just now."
```

```
Before inserting something into a liquid source: say
"[The noun] would get lost and never be seen again."
instead.
```

```
Part 3 - Scenario
```

The Lakeside is a room. The Lakeside swing is an enterable supporter in the Lakeside. "Here you are by the lake, enjoying a summery view."

The glass is a fluid container carried by the player. The liquid of the glass is absinthe. The current volume of the glass is 0.8 fl oz.

The pitcher is a fluid container in the Lakeside. The fluid capacity of the pitcher is 32.0 fl oz. The current volume of the pitcher is 20.0 fl oz. The liquid of the pitcher is absinthe.

The lake is a liquid source. It is in the Lakeside.

The player wears a bathing outfit. The description of the bathing outfit is "Stylishly striped in blue and white, and daringly cut to reveal almost all of your calves, and quite a bit of upper arm, as well. You had a moral struggle, purchasing it; but mercifully the lakeshore is sufficiently secluded that no one can see you in this immodest apparel."

Instead of taking off the outfit: say "What odd ideas come into your head sometimes!"

Test me with "fill glass / empty absinthe into lake / fill glass / swim / drink lake / drink / x water / x lake".

Chapter 18: Activities

; *§18.20. Printing the announcement of light*; *§18.21. Printing the name of a dark room*; *§18.22. Printing*

Contents of *Writing with Inform*
Chapter 17: Understanding
Chapter 19: Rulebooks
Indexes of the examples

§18.1. What are activities?

It is poor form to define with negatives, but the first thing to say about activities is that they are *not* actions. This needs saying because Inform often seems to treat them as if they are, by allowing us to write rules like so:

```
Before printing the name of a woman, say "Ms ".
```

With this rule in place, someone called "Daphne" will always be described as "Ms Daphne", and so on. The language looks as if we were imposing a rule on an action called "printing the name of", but there is no such action: instead, it is an "activity". To spell out the difference:

```
An action is a simulated task for the fictional protagonist.

An activity is a real task for the computer program doing the
simulation.
```

Activities allow us to influence or change some of the standard habits of Inform, using rules as flexible and powerful as those applicable to actions, though activities are in several ways simpler and easier.

Start of Chapter 18: Activities
Back to Chapter 17: Understanding: §17.22.

Precedence
Onward to §18.2. How activities work

Example 334: ★ **Ant-Sensitive Sunglasses** What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

---

§18.2. How activities work

All activities start, continue for a while and then finish: however, no activity ever runs on for more than a single turn. Several activities can be going on at the same time. For instance, suppose the following is printed as part of the description of a grocery:

```
You can see a banana, an apple and a star-fruit here.
```

At the moment when Inform prints "apple", two activities are under way: "listing contents of the Grocery", and "printing the name of the apple". The sequence of events was in fact:

```
say "You can see "
start listing contents of the Grocery
    say "a "
    start printing the name of the banana
        say "banana"
    finish printing the name of the banana
    say ", an "
    start printing the name of the apple
        say "apple"
    finish printing the name of the apple
    say " and a "
    start printing the name of the star-fruit
        say "star-fruit"
    finish printing the name of the star-fruit
finish listing contents of the Grocery
```

```
say " here."
```

The golden rule is: if activity B starts during activity A, it must also finish during activity A.

If we ever need to find out, we can always test:

```
if the printing the name activity is going on, ...
if the printing the name activity is not going on, ...
```

but as we shall see, it's usually simpler to attach "while printing the name" provisos to rules.

Start of Chapter 18: Activities
Back to §18.1. What are activities?
Onward to §18.3. Rules applied to activities

§18.3. Rules applied to activities

The activity "printing the name of something" is the process of printing up the name of something on screen: ordinarily, this means saying the text in its "printed name" property.

As with actions, rules can be attached to activities which change or augment what would normally happen. In fact the situation is simpler, because (unlike an action) an activity almost always finishes, so we almost always do reach its "after" stage. There are also only three rulebooks attached to an activity, as compared with the six affecting an action.

The three rulebooks for printing the name are called "before printing the name", "for printing the name" and "after printing the name", and this is the general pattern. What happens is:

```
1. All "before printing the name of" rules are considered;
2. The most specific, applicable "rule for printing the name of" is
considered;
3. All "after printing the name of" rules are considered.
```

Whereas an action's later stages never take place if an early stage ends

unexpectedly, an activity always goes through all three of its stages. Invoking the word "instead" in a before rule for an action will terminate not only the before rules but the whole action: the same thing for an activity will only terminate the before rules, and the for and after rules will take place as usual.

The actual task is usually carried out by one single rule tucked into the back of the "for..." rulebook: it is the rule for printing the name of whatever is concerned, hence the name. Inform's standard activities are all of this pattern: they start out with no "before" or "after" rules, and just one "for" rule.

Why the part about an activity only "almost always" finishing? One reason is that the story might end during it; but another is that it's possible, though uncommon, to abandon an activity partway. Very few of the activities supplied with Inform ever do this, and those that do are noted in the sections which follow.

Start of Chapter 18: Activities
Back to §18.2. How activities work
Onward to §18.4. While clauses

§18.4. While clauses

Rules applied to actions can become baroque ("after going through a door in the presence of an animal when -" and so on and so forth), but activities are again simpler: they only have one possible clause attached, which is called "while". For instance, the following would provide a fairly sledgehammer hint that the sack should not lightly be thrown away:

```
The sack is a player's holdall. The sack is carried. Rule for printing
the name of the sack while the sack is not carried: say "your
abandoned sack".
```

Any condition can be given after the "while", and we can also specify that another activity has to be going on. Thus:

```
Rule for printing the name of the lemon sherbet while listing contents:
say "curious sort of lemon sherbet sweet".
```

This nicely distinguishes between contexts where it's appropriate to be more verbose, and where it isn't. Thus:

```
You can see a teaspoon and a curious sort of lemon sherbet sweet
here.
> TAKE ALL
teaspoon: Taken.
lemon sherbet: Taken.
```

Start of Chapter 18: Activities

Back to §18.3. Rules applied to activities

Onward to §18.5. New activities

§18.5. New activities

Activities are all about influencing the standard mechanisms which Inform uses, so it might at first seem that there is no need to create new activities: but on further reflection, quite a lot of the writing of interactive fiction involves creating new and systematic ways to do things, and as soon as we have a general rule, we will want to have exceptions. Inform therefore allows us to create our own activities, giving us ways to influence the operation of our own mechanisms.

There are two kinds of activity: those which relate to a specific value (usually an object but not necessarily), and those which do not. Here are some examples of activities being created:

```
Assaying is an activity.
Analysing something is an activity.
Announcing something is an activity on numbers.
```

Inform looks for the clue "something" (or "of something") after the activity's name to see if it will work on a value: so analysing and announcing will do, but assaying won't. If we don't specify a kind, Inform assumes the value will be an object, as if we had written:

```
Analysing something is an activity on objects.
```

As always in Inform, the names of activities are themselves values.

```
"assaying activity" has kind activity on nothing
"analysing activity" has kind activity on objects
"announcing activity" has kind activity on numbers
```

Creating an activity is like creating an action: it automatically makes new rulebooks - "before analysing", "for analysing" and "after analysing" - but they start out empty, so the activity does nothing yet. Just as it does for rulebooks, Inform defines the adjectives "empty" and "non-empty" for activities to test this state:

```
if the analysing activity is empty, ...
```

will be true only when all three of its rulebooks are empty.

A newly created activity never happens unless we take steps to make it do so. We can make an activity happen at any time by writing phrases like so:

carry out the (activity) **activity**

This phrase carries out the given activity, which must be one not applying to any value. Example:

```
carry out the assaying activity;
```

carry out the (activity on values) **activity with** (value)

This phrase carries out the given activity, which must apply to a kind of value matching the one supplied. Example:

```
carry out the analysing activity with the pitchblende;
carry out the announcing activity with the score;
```

To make the activity do something useful, we need to put a rule into its "for" rulebook:

```
Rule for announcing a number (called N): say "Ladies and
gentlemen, [N]."
```

```
The last for assaying rule:
    say "Professionally, you cast an eye around mineral deposits
nearby, noticing [list of rocks in the location]."
```

"The last" is a technicality about rulebooks (see the next chapter) which, put briefly, guarantees that this rule comes last among all possible "for assaying" rules. This is good form because the whole point of an activity is to make it easy for further rules to interfere - so we deliberately hang back to last place, giving precedence to anybody else who wants it.

The "for" rulebook is one where rules stop the activity, by default, when they take effect - in the same way that the "instead" rules stop actions by default. If this causes problems, we can use:

continue the activity

This phrase should be used only in rules in activity rulebooks. It causes the current rule to end, but without result, so that the activity continues rather than stopping as a result of the rule. This is useful for rulebooks (like the "for" rulebook of an activity) where the default is that a rule does stop the activity.

Activities are more useful than they first appear. Every new one provides a context which other activities can observe. We could, for instance, define

```
Rule for printing the name of a rock while assaying: ...
```

so that during assays more technical names are used.

Start of Chapter 18: Activities

Back to §18.4. While clauses

Onward to §18.6. Activity variables

Example 335: ★★ **AARP-Gnosis** An Encyclopedia set which treats volumes in the same place as a single object, but can also be split up.

Example 336: ★★★

Aftershock Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times.

Example 337: **Crusoe** Adding a "printing the description of something" activity.

---

§18.6. Activity variables

Just as actions can have variables, which are created when the action starts and disappear when it finishes, so activities can also have variables. They are visible to the rules for that activity, and nowhere else. (If the activity should happen a second time within its first run, that second occurrence gets its own copy of the variable, leaving the original untouched.)

Typically it will be useful to set a variable to some default value at the "before" stage, calculate some interesting value for it in the "for" stage, and make use of the outcome during the "after" stage. For instance:

```
Analysing something is an activity. The analysing activity has a text
called first impression. Instead of examining something (called the
sample), carry out the analysing activity with the sample.

Before analysing: now the first impression is "unremarkable".

Rule for analysing someone: now the first impression is "living
tissue".

After analysing something (called the sample):
    say "Your professional opinion of [the sample] is that it is
        [first impression]."
```

Start of Chapter 18: Activities

§18.7. Beginning and ending activities manually

If we have declared a new activity, like "analysing", the normal way to make it happen would be to write

```
carry out the analysing activity with the pitchblende;
```

which goes through the whole machinery of rules - before, for, after - and then resumes, the activity having started, taken place and come to an end.

But there are times when it is not convenient to write a suitable "for ..." rule, or where we need more control, and do not wish to hand the whole business over to a single phrase. For such times we are allowed to write:

begin the (activity) **activity**

This phrase causes the named activity to become active, and runs its "before" rulebook. The activity must be one which applies to nothing. Example:

```
begin the assaying activity;
```

In all cases a matching "end the ... activity" or else "abandon the ... activity" phrase must be reached.

begin the (activity on values) **activity with** (value)

This phrase causes the named activity to become active, and runs its "before" rulebook. The activity must be one which applies to a value of a matching kind. Example:

```
begin the analysing activity with the pitchblende;
```

In all cases a matching "end the ... activity with ..." or else "abandon the ... activity with..." phrase must be reached.

And when we are done:

end the (activity) **activity**

This phrase runs the "after" rulebook of the activity and then causes it to become inactive. The activity must be one which applies to nothing. Example:

```
end the assaying activity;
```

This must only happen to match an earlier "begin the ... activity" phrase.

end the (activity on values) **activity with** (value)

This phrase runs the "after" rulebook of the activity and then causes it to become inactive. The activity must be one which applies to a value of a matching kind. Example:

```
end the analysing activity with the pitchblende;
```

This must only happen to match an earlier "begin the ... activity with..." phrase.

So the usual structure is like so:

```
begin the analysing activity with the pitchblende;
...
end the analysing activity with the pitchblende;
```

This time the activity is ongoing throughout as many phrases as we care to write between the "begin" and "end". The before rules are considered at the time of the "begin ..." phrase; the after rules at the "end ...".

What, then, of the "for" rules? In the above setup, they would simply be ignored. But we can make them effectual thus

```
begin the analysing activity with the pitchblende;
...
if handling the analysing activity with the pitchblende:
    ...
...
end the analysing activity with the pitchblende;
```

We place the activity's normal behaviour inside the "if"; the condition, "if handling...", is true only if no rule has intervened. This means that we (or other authors using our activity) can create their own for rules to substitute here. If we elsewhere write

```
Rule for handling the analysing activity with the pitchblende when the
player is not sober:
    say "You can't seem to focus."
```

that rule will intervene and take the place of whatever we have placed inside the condition.

if handling (activity) **activity:**

This should be used only where the given activity has been started with "begin ..." and will be finished with "end ...". It runs the "for" rules for the activity, and then comes out true if none of those for rules intervened in the handling of that activity. (The activity must be one which doesn't apply to any value.)

if handling (activity on values) **activity with** (value)**:**

This should be used only where the given activity has been started with "begin ..." and will be finished with "end ...". It runs the "for" rules for the activity, and then comes out true if none of those for rules intervened in the handling of that activity. (The given value must be the one it is being applied to.)

It is also legal to force an early end to an activity with:

abandon the (activity) **activity**

This phrase ends an activity at once (without consulting any further rulebooks, including its "after" rulebook). It can only be used with an activity which has had its "begin" but not yet its "end" phrase; it is a drastic remedy best taken only if it is clear that circumstances have changed so that the activity now seems inappropriate. It must not be used during one of the rules for the activity: it can only be used between the begin and for stages, or between the for and end stages.

```
abandon the assaying activity;
```

abandon the (activity on values) **activity with** (value)

This phrase ends an activity at once (without consulting any further rulebooks, including its "after" rulebook). It can only be used with an activity which has had its "begin" but not yet its "end" phrase; it is a drastic remedy best taken only if it is clear that circumstances have changed so that the activity now seems inappropriate. It must not be used during one of the rules for the activity: it can only be used between the begin and for stages, or between the for and end stages.

```
abandon the analysing activity with the pitchblende;
```

We need to follow three golden rules: all activities must end, they must never last longer than a turn, and if activity B starts during activity A then it must also finish during activity A. We must also be careful to make sure that if an activity applies to something, then it begins and ends with the same something (the pitchblende, in the above example).

Start of Chapter 18: Activities
Back to §18.6. Activity variables
Onward to §18.8. Introduction to the list of built-in activities

§18.8. Introduction to the list of built-in activities

Activities tend to be about process, rather than outcome. Many of the things Inform does - printing up lists of items, reading commands from the keyboard, and so on - are done as activities, because that way the process can be nudged a little. Too many works of interactive fiction betray their mechanical nature by making it visible that the general machinery being used does not quite seem natural for this or that situation. Activities enable us to add the many graceful touches which avoid that: which contribute nothing to a work, and also everything.

The rest of this chapter covers every activity built in to Inform, with one section for each. It is intended primarily for reference, but may be worth skimming through at a first reading, to give a sense of the possibilities.

Start of Chapter 18: Activities

Back to §18.7. Beginning and ending activities manually

Onward to §18.9. Deciding the concealed possessions of something

§18.9. Deciding the concealed possessions of something

1. When it happens. Frequently - whenever Inform needs to check whether something is visible or not. Nothing should be printed, and the activity needs to run quickly, so it should not (for instance) calculate best routes through complicated maps before getting an answer.

2. The default behaviour. There is no concealment. The ordinary rules still apply, though: the contents of a closed opaque container are invisible because there is a barrier in the way which cannot be seen through, even though nobody is "concealing" anything.

3. Examples. To repeat a number of brief examples given at the end of Chapter 3, where this activity made an early appearance:

```
Rule for deciding the concealed possessions of the Cloaked Villain: if
the particular possession is the sable cloak, no; otherwise yes.
```

```
The coin is in the Roman Villa. The face and inscription are parts of
the coin. Rule for deciding the concealed possessions of the coin: if
the coin is carried, no; otherwise yes.
```

The value "particular possession" is the one whose concealment is in question, of course. We can ignore this if someone is invariably secretive:

```
Rule for deciding the concealed possessions of the furtive ghost: yes.
```

In general a rule for deciding the concealed possessions of something will decide "yes" if finishes without making a decision, but it's better style to write such a rule in such a way that it always makes a decision.

Start of Chapter 18: Activities

Back to §18.8. Introduction to the list of built-in activities

Onward to §18.10. Printing the name of something

Example 338: ★ **Hays Code** Clark Gable in a pin-striped suit and a pink thong.

§18.10. Printing the name of something

1. When it happens. Whenever the name of a thing or room is printed.

2. The default behaviour. For items other than the current player, the "printed name" property is printed out; but for the current player, "you" or "yourself" is printed. (That doesn't necessarily mean that the "printed name" of the player is never used. Suppose there are two people, Alice and Bob, and the narrative switches between them: when Alice is the player, she appears as "yourself" but Bob is "Bob"; but when Bob is the player, he is "yourself" and Alice is "Alice".)

3. Examples. (a) A pen which is described differently in inventories:

```
Rule for printing the name of the pen while taking inventory: say
"useful pen".
```

"Taking inventory" is a condition which is true if that's the current action and not otherwise, so the effect is that the pen is called "a useful pen" only in inventory listings. "While looking" is a similarly useful one.

(b) Italicising the names of novels:

```
A novel is a kind of thing. Dr Zhivago and Persuasion are novels.
Before printing the name of a novel, say "[italic type]". After printing
the name of a novel, say "[roman type]".
```

(c) Telling the time:

```
After printing the name of the wrist watch while taking inventory: say "
(time: [the time of day])".
```

(d) Merging containers with their contents:

```
Rule for printing the name of the bottle while not inserting or
removing:
    if the bottle contains sand, say "bottle of sand";
    otherwise say "empty bottle";
    omit contents in listing.
```

This example makes use of a special phrase:

omit contents in listing

This phrase changes the form of an inventory listing, room description, etc., so that it will simply list "a bottle of sand" or "an empty bottle", rather than "a bottle (in which is sand)" or "a bottle (which is empty)". It should be used only when the listing is imminent, and does not have permanent effect.

The clause about not inserting or removing is to prevent messages like "You put the sand in the bottle of sand.", where it's confusing to refer to the bottle as anything other than "the bottle".

Start of Chapter 18: Activities

Back to §18.9. Deciding the concealed possessions of something

Onward to §18.11. Printing the plural name of something

Example 339: ★ **Shipping**

Trunk A box of baking soda whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

Example 340: ★★ **Trachypachidae Maturin 1803** Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

Example 341: ★★★★ **Chronic Hinting Syndrome** Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

---

§18.11. Printing the plural name of something

1. When it happens. Only when a group of identical items is present in the same place, and are being described jointly with text like "You can see five gold rings here." The activity happens after "five" and before "here." (See the activity "printing a number of something" if the whole phrase needs to be altered.)

2. The default behaviour. The plural name - in this case "gold rings" - is printed out.

3. Examples. (a) Suppose we want to emphasise how nice it is to have more than one gold ring:

```
Rule for printing the plural name of a gold ring: say "gleaming gold
rings".
```

(b) If the number needs changing as well, it's necessary to use the "printing a number of something" activity instead.

Start of Chapter 18: Activities

Back to §18.10. Printing the name of something

Onward to §18.12. Printing a number of something

Example 342: ★★ **Hudsucker Industries** Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

§18.12. Printing a number of something

1. When it happens. Only when a group of identical items is present in the same place, and are being described jointly with text like "You can see five gold rings here." The activity prints the "five gold rings" part. The variable "listing group size" contains the number, which in this example would be 5, and is always at least 2.

2. The default behaviour. The number of items is printed, in words ("five") and then the "printing the plural name" activity is run ("gold rings").

3. Examples. (a) Using this activity is for perfectionists, because the normal behaviour is almost always fine. Still:

```
Rule for printing a number of blocks when the listing group size is 3:
say "all three blocks".
```

(b) Or perhaps:

```
Rule for printing a number of ants: say "altogether [listing group size
in words] ants".
```

(c) If the only part needing variation is the plural name, it's simpler and tidier to use the "printing the plural name of something" activity instead.

Start of Chapter 18: Activities

Back to §18.11. Printing the plural name of something
Onward to §18.13. Listing contents of something

Example 343: **Prolegomena** Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

---

§18.13. Listing contents of something

1. When it happens. When taking inventory, the list is produced by the activity "listing contents of yourself"; when looking, a list of items which do not deserve their own paragraphs is produced by "listing contents of" the location.

And when it doesn't happen. (a) If the Storage Room contains a sideboard and an open shoe box, then "listing contents of the Storage Room" is used to produce the part of the room description mentioning sideboard and box. But if the box in turn contains a pair of brogues, then "listing contents of the shoe box" is not used to say that part. So this works:

```
Rule for printing the name of the brogues while listing contents of a
room: ...
```

But this won't affect room descriptions:

```
Rule for printing the name of the brogues while listing contents of the
shoe box: ...
```

(b) The activity also doesn't happen when, for instance, "[a list of animals]" is printed, because that isn't a list of the contents of any room or location.

2. The default behaviour. The list is printed out.

3. Examples. (a) We have already seen that it can be elegant to elaborate on a description in the context of a list. Here we add "discarded" to a sweet wrapper which is found on the ground.

```
Rule for printing the name of the wrapper while listing contents of a
room: say "discarded sweet wrapper".
```

(b) Lists can be considerably shortened and tidied up if similar items are grouped together. We do this by specifying what should be grouped together before listing contents, using the special phrase "group ... together":

```
Utensil is a kind of thing. The knife, the fork and the spoon are
utensils. Before listing contents: group utensils together as "utensils".
```

The result will be, say, "two utensils (knife and spoon)", if both are found in the same place.

(c) We can less obtrusively group items together like so:

```
Before listing contents while taking inventory: group utensils together.
```

Three special phrases exist for this kind of list organisation:

group (description of objects) **together**

This phrase causes the objects described to be listed together in a single item as part of an inventory or room description. The effect is temporary, and the phrase should only be used when this list is imminent. Example:

```
Utensil is a kind of thing. The knife, the fork and the spoon
are utensils. Before listing contents: group utensils together.
```

This might produce the list item "fork and spoon".

group (description of objects) **together giving articles**

This phrase causes the objects described to be listed together in a single item as part of an inventory or room description, but giving each individual item its indefinite article. The effect is temporary, and the phrase should only be used when this list is imminent. Example:

```
Utensil is a kind of thing. The knife, the fork and the spoon
are utensils. Before listing contents: group utensils together
giving articles.
```

This might produce the list item "a fork and a spoon".

group (description of objects) **together as** (text)

This phrase causes the objects described to be listed together in a single item as part of an inventory or room description, summarised with the given text. The effect is temporary, and the phrase should only be used when this list is imminent. Example:

```
Utensil is a kind of thing. The knife, the fork and the spoon
are utensils. Before listing contents: group utensils together
as "utensils".
```

This might produce the list item "two utensils (fork and spoon)".

Start of Chapter 18: Activities

Back to §18.12. Printing a number of something

Onward to §18.14. Grouping together something

Example 344: **Unpeeled** Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

§18.14. Grouping together something

1. When it happens. Only while listing contents, and only when a collection of items to be grouped together is reached. This in turn happens only if a "before listing contents" rule has chosen it (see previous section). The first item in the group is the one to which the activity formally applies. The variable "listing group size" gives the

number of items grouped together in this way.

2. The default behaviour. The items grouped together are printed in an English phrase, such as "egg, chicken and farmer". In particular, they are not split onto separate lines even if the rest of the list is. (See previous section.)

3. Examples. (a) Here are Scrabble pieces which are described as "the tile W from a Scrabble set" or similar outside of lists, but which, when they turn up together in lists, are rolled together into "the tiles A, B and D from a Scrabble set".

```
A Scrabble piece is a kind of thing. The X, the Y and the Z are
Scrabble pieces.

Before listing contents: group Scrabble pieces together.

Before printing the name of a Scrabble piece while not grouping
together, say "tile ". After printing the name of a Scrabble piece while
not grouping together, say " from a Scrabble set".

Before grouping together Scrabble pieces, say "the tiles ". After
grouping together Scrabble pieces, say " from a Scrabble set".
```

(b) Maybe we only want an abbreviated form when there are five or more tiles in one place:

```
A Scrabble piece is a kind of thing. The X, the W, the F, the Y and
the Z are Scrabble pieces in the Lounge.

Before listing contents: group Scrabble pieces together.

Before grouping together Scrabble pieces when the listing group size
is greater than 4:
    say "some [listing group size in words] tiles (".
After grouping together Scrabble pieces when the listing group size is
```

```
greater than 4:
    say ") from a Scrabble set".
```

(c) We can throw out all pretence at listing and say whatever we like, in fact:

```
Before listing contents while taking inventory: group utensils together.
Rule for grouping together utensils: say "the usual utensils".
```

Start of Chapter 18: Activities
Back to §18.13. Listing contents of something
Onward to §18.15. Issuing the response text of something

§18.15. Issuing the response text of something

1. When it happens. When Inform prints a text marked with a response letter (A), (B), (C), ..., in a rule making use of them. For example, in this rule:

```
Carry out taking inventory (this is the print empty inventory rule):
    if the first thing held by the player is nothing,
        say "[We] [are] carrying nothing." (A) instead.
```

Or, less directly,

```
let R be the print empty inventory rule response (A);
say "To be frank: [text of R].";
```

2. The default behaviour. To print the current textual value of the response, making any substitutions in the ordinary way.

3. Examples. This activity is not the best way to amend responses or make them dynamic; the whole idea of responses is that they can be changed just as if they were text variables. This activity should be used only if it's important to amend blocks of responses in some systematic way.

(a) With that said, some interesting effects can be achieved. This is a way to see which responses are being printed, for example:

```
Before issuing the response text of a response (called R): say "[R]: ".
```

whence:

```
>WAIT
standard report waiting rule response (A): Time passes.
```

(b) And this intercepts the activity in order to re-run it in each of the six viewpoints. (Note the way a variable is used to prevent the rule from applying to all of those re-runs as well.)

```
The response inhibition is initially false.


Rule for issuing the response text of a response (called R) when the
response inhibition is false:
    now the response inhibition is true;
    let the current viewpoint be the story viewpoint;
    repeat with P running through narrative viewpoints:
        now the story viewpoint is P;
        say "[P]: [text of R][command clarification break]";
    now the story viewpoint is the current viewpoint;
    now the response inhibition is false.
```

With that in place,

```
>EAST
first person singular: I can't go that way.
second person singular: You can't go that way.
third person singular: He can't go that way.
first person plural: We can't go that way.
second person plural: You can't go that way.
third person plural: They can't go that way.
```

Start of Chapter 18: Activities

Back to §18.14. Grouping together something

Onward to §18.16. Printing room

§18.16. Printing room description details of something

1. When it happens. When an item is listed in the miscellaneous collection of items present in a room (the ones which do not deserve their own paragraphs): this is normally the last paragraph of a room description.

2. The default behaviour. A bracketed piece of extra information is added for certain items such as containers:

```
You can also see Po and a cage (empty) here.
```

The " (empty)" (note initial space) was added by this activity. (Note that this activity is not responsible for describing further items visible because of the item in question: that is, it does not print the text such as "(in which is a notepad)" which would appear if there were contents. If we want to remove such text, we should use "omit contents in listing": see the activity "for printing the name of something".)

3. Examples. (a) To get rid of such addenda entirely, try:

```
Rule for printing room description details: stop.
```

(b) To add a new form of addendum:

```
Rule for printing room description details of a person:
    say " (at last, someone to talk to)" instead.
```

If both examples (a) and (b) are in place at once, we might now read:

```
You can also see Po (at last, someone to talk to) and a cage here.
```

Start of Chapter 18: Activities
Back to §18.15. Issuing the response text of something
Onward to §18.17. Printing inventory details of something

Example 346: ★ **Rules of Attraction** A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

---

§18.17. Printing inventory details of something

1. When it happens. When an item is listed in an inventory of items carried by the player.

2. The default behaviour. A bracketed piece of extra information is added for certain items such as containers:

```
a flaming branch (providing light)
```

The " (providing light)" (note initial space) was added by this activity.

3. Examples. (a) To get rid of such addenda entirely, try:

```
Rule for printing inventory details: stop.
```

(b) To add a new form of addendum:

```
Rule for printing inventory details of something edible:
    say " (yummy!)[run paragraph on]".
```

Start of Chapter 18: Activities
Back to §18.16. Printing room

description details of something

Onward to §18.18. Printing a refusal to act in the dark

§18.18. Printing a refusal to act in the dark

1. When it happens. When an action which requires light is tried, and the visibility rules decide that not enough light is present.

2. The default behaviour. To print "It is pitch dark, and you can't see a thing."

3. Examples. (a) This might do for some twilit, penumbral room:

```
Rule for printing a refusal to act in the dark: if we are examining
something, say "It's not totally dark here, perhaps, but certainly too
dim for close-up examination of anything." instead.
```

Start of Chapter 18: Activities

Back to §18.17. Printing inventory details of something

Onward to §18.19. Printing the announcement of darkness

Example 347: ★★★ **Zorn of Zorna** Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

§18.19. Printing the announcement of darkness

1. When it happens. Inform frequently calculates to see if the player is in light or darkness: this activity happens on the change from light to darkness.

2. The default behaviour. To print "It is now pitch dark in here!".

3. Examples. (a) The most obvious use is to change the text:

```
Rule for printing the announcement of darkness: say "Ooh-er! It's
now very nearly pitch dark in here." instead.
```

(b) But we could also use this activity for sneakier purposes, silently moving things around:

```
Before printing the announcement of darkness: now all of the
gremlins are in the kitchen.
```

(c) A special description for occasions when the player has climbed into a container and shut it (so that the darkness is the result of his own actions, rather than some external circumstance):

```
Rule for printing the announcement of darkness when closing a
container which contains the player:
    say "Congratulations: now you can't see a thing." instead.
```

Start of Chapter 18: Activities

Back to §18.18. Printing a refusal to act in the dark

Onward to §18.20. Printing the announcement of light

§18.20. Printing the announcement of light

1. When it happens. Inform frequently calculates to see if the player is in light or darkness: this activity happens on the change from darkness to light.

2. The default behaviour. To try the looking action, which usually prints up a room description.

3. Examples. (a) Perhaps the player is initially too disoriented to look around in any coherent way:

```
Rule for printing the announcement of light in the Dazzling Temple:
say "You are almost blinded by the suffusion of white light, and have
spots before your eyes." instead.
```

Start of Chapter 18: Activities

Back to §18.19. Printing the announcement of darkness

Onward to §18.21. Printing the name of a dark room

§18.21. Printing the name of a dark room

1. When it happens. When looking in darkness, or writing the (default) status line in darkness.

2. The default behaviour. To print "Darkness".

3. Examples. (a) One might modify the darkness with some adjective:

```
Before printing the name of a dark room, say "Near ".
```

(Note that this activity does not come in different forms for different dark rooms: the wording is fixed at "printing the name of a dark room", and we are not allowed to substitute particular dark rooms or assign a "(called ...)" onto the mention of the dark room.)

Start of Chapter 18: Activities

Back to §18.20. Printing the announcement of light

Onward to §18.22. Printing the description of a dark room

§18.22. Printing the description of a dark room

1. When it happens. When looking in darkness.

2. The default behaviour. To print "It is pitch dark, and you can't see a thing."

3. Examples. (a) A simple variation of wording:

```
Rule for printing the description of a dark room: say "Your eyes can
```

```
barely make anything out." instead.
```

(b) More stylishly,

```
Rule for printing the description of a dark room: try listening instead.
```

which produces, for instance,

```
Darkness
You hear nothing unexpected.
```

(Note that this activity does not come in different forms for different dark rooms: the wording is fixed at "printing the description of a dark room", and we are not allowed to substitute particular dark rooms or assign a "(called ...)" onto the mention of the dark room.)

Start of Chapter 18: Activities

Back to §18.21. Printing the name of a dark room

Onward to §18.23. Constructing the status line

Example 348: **Hohmann Transfer** Changing the way dark rooms are described to avoid the standard Inform phrasing.

Example 349: **Four Stars 1** An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead.

---

§18.23. Constructing the status line

1. When it happens. Just before input is accepted from the keyboard, Inform constructs a "status line" at the top of the window which is normally displayed in reverse colours (white on black instead of black on white, say).

2. The default behaviour. Makes the status line up out of two pieces, the "left hand status line" and the "right hand status line". Since these can freely be changed, note that the status line is already very customisable without using rules applied to this activity.

3. Examples. (a) The most useful thing about this activity is that it allows us to vary descriptions in the status line. This is especially helpful to abbreviate unduly long room names, which might not otherwise fit:

```
The Temple Of A Thousand Mightily Peeved Deities is a room. Rule
for printing the name of the Temple while constructing the status line:
say "Temple".
```

(b) Again, it's usually not necessary to apply activity rules to this, but occasionally amusing effects are possible if we do:

```
The blindfold is wearable and carried. Rule for constructing the status
line while the blindfold is worn: do nothing.
```

Start of Chapter 18: Activities

Back to §18.22. Printing the description of a dark room

Onward to §18.24. Writing a paragraph about

Example 350: ★ **Ways Out** A status line that lists the available exits from the current location.

Example 351: ★★ **Guided Tour** A status line that lists the available exits from the current location, changing the names of these exits depending on whether the room has been visited or not.

§18.24. Writing a paragraph about

1. When it happens. Just *before* writing a paragraph about some item in a room description.

2. The default behaviour. Is to do nothing. However, if a rule is supplied which prints something up, then this replaces the paragraph which would otherwise have been printed. Moreover, any items whose names are said in the course of this rule - for instance, by being listed - are then excluded from the remainder of the room description, because they are considered as having been described sufficiently already.

Warning: because we often want a "for" rule for this activity to make some calculation and then possibly choose to do nothing (see the example "Otranto"), Inform suppresses the usual paragraph not when a "for" rule took effect but when it detected a paragraph having been printed. This can get confused if a text substitution affecting paragraph breaks, say "[line break]", is within the final "say" of a "for writing a paragraph about" rule.

3. Examples. (a) This is a neat way to wrap several things together into the same paragraph:

```
Rule for writing a paragraph about Mr Wickham:
    say "Mr Wickham looks speculatively at [list of women in the
location]."
```

because now "Mr Wickham looks speculatively at Velma and Daphne" will now prevent the appearance of the subsequent text "You can also see Velma and Daphne."

Inform keeps track of which objects have already been named with an either/or property called "mentioned", which it assigns whenever the name of an object has been automatically printed. So in this case, Velma and Daphne are now mentioned. Note "automatically printed", though: if the text printed had just been "Mr Wickham looks speculatively at Velma and Daphne", rather than the text-substitution list used above, then Inform would not know that Velma and Daphne have been described.

If we ever need to override this - say, we want to list all the women but make sure that Velma gets another paragraph anyway - we could change Velma to unmentioned again after the listing.

Start of Chapter 18: Activities

Back to §18.23. Constructing the status line

Onward to §18.25. Listing nondescript items of something

Example 352: ★ **Reflections** Emphasizing the reflective quality of shiny objects whenever they

are described in the presence of the torch.

Example 353: ★★ **Emma** Social dynamics in which groups of people form and circulate during a party.

Example 354: ★★★★ **Air Conditioning is Standard** Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

---

§18.25. Listing nondescript items of something

1. When it happens. This activity prints up the also-ran paragraph at the end of a room description. These are nondescript items because they don't merit paragraphs of their own: if, as sometimes happens, there are none in the room, then no such paragraph is printed and this activity does not happen. (So to add a further paragraph to a room description, a simpler "after looking" rule should be used, not an "after listing nondescript items" rule.)

2. The default behaviour. The paragraph ordinarily reads as "You can also see a cask and a clock." or similar. Before the activity begins, those objects which are nondescript - in this case the cask and the clock - are given the property of being "marked for listing".

If it turns out that nothing is marked for listing, because of before rules like the one in the example below, then nothing is printed and **the activity is abandoned**, so that the rules for and after are never reached.

3. Examples. (a) Promoting something out of the nondescript category, by unmarking it.

```
Before listing nondescript items:
    if the watch is marked for listing:
        say "The watch catches your eye.";
```

```
now the watch is not marked for listing.
```

(b) Changing the normal phrasing of the paragraph. Note that we can also change the listing style; the one below is the default.

```
Rule for listing nondescript items of the Distressingly Messy Room:
    say "Strewn carelessly on the floor";
    list the contents of the Distressingly Messy Room, as a sentence,
        tersely, listing marked items only, prefacing with is/are,
        including contents and giving brief inventory information;
    say "."
```

Start of Chapter 18: Activities

Back to §18.24. Writing a paragraph about

Onward to §18.26. Printing the locale description of something

Example 355: ★ **Rip Van Winkle** A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

Example 356: ★★ **Happy Hour** Listing visible characters as a group, then giving some followup details in the same paragraph about specific ones.

Example 357: ★★ **The Eye of the Idol** A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

---

§18.26. Printing the locale description of something

1. When it happens. A "locale description" is Inform jargon for the part of a room description which catalogues the visible items in the room. When looking, Inform will normally print the description of the room itself, followed by a locale description for the room. But if the player is in a cage in the room, there will be two locale descriptions: one for the room, then another for the cage. This activity is used to write the locale description for a single domain, and the "something" can be either a room, an enterable container, or an enterable supporter.

2. The default behaviour. Is quite complicated, and is written up in full in the typeset form of the Standard Rules downloadable from the Inform website. Briefly, though: we first run the "choosing notable locale objects" activity to find out what ought to be mentioned here. That assembles a list of things to mention, sorted into priority order. Items with priority 1 go first, then those with priority 2, and so on. The "printing a locale paragraph" activity is run for each, and in practice that usually hands the job over to "writing a paragraph about". Sometimes a paragraph will indeed be written, but not always. Sometimes there is nothing interesting to say, and an item is left until a final, single paragraph which gathers up the leftovers ("You can also see a scarlet fish, a harmonium and a kite here."), the printing of which is done by the "listing nondescript items of" activity. As soon as any item picks up the either/or property "mentioned", by having its name printed, it is struck out so that it will not appear subsequently, whatever its priority.

3. Examples. As general advice: if the effect wanted can be got using "writing a paragraph about" and "listing nondescript items of" alone, use those; if it's necessary to meddle further, use "choosing notable locale objects" and "printing a locale paragraph" to alter the normal processes; use the all-powerful "printing the locale description" activity only when the whole process needs to be altered, not the item-by-item workings.

(a) In the Very Misty Moorlands, nothing on the ground can ordinarily be seen through the swirling mist, so the locale description is suppressed entirely:

```
Rule for printing the locale description of the Very Misty Moorlands:
    say "Mist coils around your feet, thick as a blanket. You cannot
even see the ground you walk upon." instead.


Report taking something in the Very Misty Moorlands:
    say "You grope blindly in the mist and pick up [the noun]." instead.
```

(b) Here we take the chance to insert an additional paragraph into the locale description. This does relate to an item which might be described later, but where the

player doesn't know that:

```
The Horological Workshop is a room. The marble table is fixed in
place in the Workshop.

The parcel is a closed opaque container on the marble table. The
alarm clock is a device in the parcel. The alarm clock is switched on.

Before printing the locale description of a room (called the locale):
    if the locale encloses the alarm clock and the alarm clock is
switched on, say "A faint ticking noise can be heard."
```

Start of Chapter 18: Activities

Back to §18.25. Listing nondescript items of something

Onward to §18.27. Choosing notable locale objects for something

Example 358: ★ **Priority Lab** A debugging rule useful for checking the priorities of objects about to be listed.

§18.27. Choosing notable locale objects for something

1. When it happens. See "printing the locale description". This activity is expected to decide which items ought to be mentioned in a locale description for a given room, enterable container or enterable supporter, and to give each item a priority, which is a number ranging upwards from 1 (which is the top priority). The lower the priority number, the earlier the mention, or at least, the earlier the opportunity to be mentioned: it's up to other activities whether to give it a paragraph of its own or not. This activity only makes something a candidate, and decides what order the candidates will be tried in.

2. The default behaviour. By default, this activity contains only the "standard notable locale objects rule". This chooses exactly those items directly contained by the locale, assigning all of them priority 5. Note that this includes scenery, and other probably

unwanted items - those will be excluded later.

3. Examples. (a) In the Misty Moorlands, only large items on the ground are visible through the mist:

```
A thing can be large or small. A thing is usually small. The stepladder
is a large thing in the Misty Moorlands.

Rule for choosing notable locale objects for the Misty Moorlands:
    repeat with item running through large things in the Misty
Moorlands:
        set the locale priority of the item to 5.

Report taking a small thing in the Misty Moorlands:
    say "You grope blindly in the mist and pick up [the noun]." instead.
```

Note the special phrase

```
set the locale priority of the item to 5;
```

which should be used only in rules for locale activities. It makes the given item a candidate and sets its priority. (Setting the priority to 0 forces an item not to be a candidate, and can thus undo the effect of previous rules.)

It's best to avoid situations where an item has a locale priority which is higher than that of something it is on top of, or inside, since this can result in an oddly-worded description.

Start of Chapter 18: Activities

Back to §18.26. Printing the locale description of something

Onward to §18.28. Printing a locale paragraph about

Example 359: **Low Light** An object that is only visible and manipulable when a bright light fixture is on.

Example 360: ★★★ **Casino Banale** Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

§18.28. Printing a locale paragraph about

1. When it happens. See "printing the locale description". By this point, the locale description process has identified a number of items as candidates to be described, and worked out a priority order. This activity is then called for each candidate in turn, starting with the highest priority items and working downwards. It can either print some text or not, and can either mark the item as "mentioned" or not: if it does, then the item won't appear subsequently in the locale description. If the activity does nothing, the item becomes "nondescript" and falls through into the final "You can also see..." paragraph, unless another rule mentions it in the mean time.

2. The default behaviour. Is provided by a sequence of seven rules:

(1) The "don't mention player's supporter in room descriptions rule" excludes anything the player is directly or indirectly standing on or, less frequently, in. The header of the room description has probably already said something like "Boudoir (on the four-poster bed)", so the player can't be unaware of this item.

(2) The "don't mention scenery in room descriptions rule" excludes scenery.

(3) The "don't mention undescribed items in room descriptions rule" excludes the player object. (It's redundant to say "You can also see yourself here.") At present nothing else in I7 is "undescribed" in this sense.

(4) The "set pronouns from items in room descriptions rule" adjusts the meaning of pronouns like IT and HER to pick up items mentioned. Thus if a room description ends "Mme Tourmalet glares

at you.", then HER would be adjusted to mean Mme Tourmalet.

(5) The "offer items to writing a paragraph about rule" gives the "writing a paragraph about" activity a chance to intervene. We detect whether it does intervene or not by looking to see if it has printed any text.

(6) The "use initial appearance in room descriptions rule" prints the "initial appearance" property of an item which has never been handled as a paragraph, if it has one.

(7) The "describe what's on scenery supporters in room descriptions rule" is somewhat controversial. It prints text such as "On the mantelpiece is a piece of chalk." for items which, like the mantelpiece, are scenery mentioned - we assume - in the main room description. (It is assumed that scenery supporters make their contents more prominently visible than scenery containers, which we do not announce the contents of.)

3. Examples. If all that's required is to supply an interesting paragraph of room description about something then it's always better to use the "writing a paragraph about" activity, not this one. This activity should only be used when the mechanism itself needs to be adjusted.

(a) The following excludes doors from room descriptions:

```
For printing a locale paragraph about a door (called the item)
    (this is the don't mention doors in room descriptions rule):
    set the locale priority of the item to 0;
    continue the activity.
```

(It's usually a good idea to "continue the activity" at the end of rules for this activity, since usually they all need to take effect for a happy outcome to the process. Here it doesn't really matter, since we were trying to stop anything from happening about the door, but it doesn't do any harm either.)

(b) Here's how to abolish what may be the most contentious rule in the whole Standard Rules:

The describe what's on scenery supporters in room descriptions rule is not listed in any rulebook.

Start of Chapter 18: Activities

Back to §18.27. Choosing notable locale objects for something

Onward to §18.29. Deciding the scope of something

Example 361: ★ **Kiwi** Creating a raised supporter kind whose contents the player can't see or take from the ground.

Example 362: ★★★ **Copper River** Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

§18.29. Deciding the scope of something

1. When it happens. "Scope" is a term of art in interactive fiction programming: it means the collection of things which can be interacted with at any given moment, which depends on who you are and where you are. Commands typed by the player will only be allowed to go forward into actions if the things they refer to are "in scope". Inform also needs to determine scope at other times, too: for instance, when deciding whether a rule conditional on being "in the presence of" something is valid. It is a bad idea to say anything during this activity.

2. The default behaviour. Is complicated: see the Inform Designer's Manual, 4th edition, page 227. Briefly, the scope for someone consists of everything in the same place as them, unless it is dark.

3. Examples. (a) We very rarely want to forbid the player to refer to things close at hand, but often want to allow references to distant ones. For instance, a mirage of something which is not present at all:

After deciding the scope of the player while in the Shrine: place the holy grail in scope.

Two different phrases enable us to place unusual items in scope:

place (object) **in scope**

This phrase should only be used in rules for the "deciding the scope of..." activity. It places the given object in scope, making it accessible to the player's commands, regardless of where it is in the model world. Examples:

```
place the distant volcano in scope;
place the lacquered box in scope, but not its contents;
```

Ordinarily if something is placed in scope, then so are its parts and (in the case of a supporter or a transparent or open container) its contents; using the "but not its contents" option we can place just the box itself in scope.

place the/-- **contents of** (object) **in scope**

This phrase should only be used in rules for the "deciding the scope of..." activity. It places the things inside or on top of the given object in scope, making them accessible to the player's commands, but it does nothing to place the object itself in scope. (It might of course be in scope anyway, and if it is then this phrase won't remove it.) Example:

```
place the contents of the lacquered box in scope;
place the contents of the Marbled Steps in scope;
```

Note that the object in question can be a room, as in this second example.

(b) Another useful device is to be able to see, but not touch, another room:

```
The Cloakroom is a room. "This is just a cloakroom, but through a
vague, misty mirror-window you can make out the Beyond." After
looking in the Cloakroom, say "In the mirror you can see [list of things
in the Beyond]."
```

```
After deciding the scope of the player while in the Cloakroom: place
```

```
the Beyond in scope.
```

```
The Beyond is a room. Johnny Depp is a man in the Beyond.
```

(This must, however, also be a mirage, as at time of writing Mr Depp is alive and as well as can be expected following the reviews of "Charlie and the Chocolate Factory".) Note that "place the Ballroom in scope" doesn't just allow the player to talk about the dancers, the chamber musicians and so forth, also allows, say, "EXAMINE BALLROOM". To get one but not the other, use "place the contents of the Ballroom in scope" or "place the Ballroom in scope, but not its contents".

(c) In darkness, the scope of someone is ordinarily restricted to his or her possessions (and body), but we can override that:

```
After deciding the scope of the player while in darkness: place the
location in scope.
```

4. A note about actions. This activity takes place during the process of understanding the player's command, when the action that will take place is not fully known. So if the player types "TAKE SHOEBOX", this activity would happen when SHOEBOX is being examined for meaning. Inform knows the action it would be taking if the current line of command grammar were to be accepted, but it does not yet know to what objects that command would be applied. That means attaching a proviso like "... while taking a container" to a rule for this activity will cause the rule to have no effect - whereas "... while taking" would be fine.

Start of Chapter 18: Activities

Back to §18.28. Printing a locale paragraph about

Onward to §18.30. Clarifying the parser's choice of something

Example 363: **Peeled** Two different approaches to adjusting what the player can interact with, compared.

Example 364: **Four Stars 2** Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

Example 365: ★★ **Ginger Beer** A portable magic telescope which allows the player to view items in another room of his choice.

Example 366: ★★ **Rock Garden** A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

Example 367: ★★★ **Stately Gardens** An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

---

§18.30. Clarifying the parser's choice of something

1. When it happens. When the player has typed an ambiguous noun reference, and Inform has made a decision about what was meant, and it matters what this decision is. (If the decision is between three identical gold coins, say, then it doesn't matter, and this activity does not take place.) There are a couple of limitations on this: the activity applies only to the first noun, and only if it's an object. So for a command like SELECT BLUE, where BLUE is a noun referring to a colour value, say, this activity isn't used. But the simple case where the activity does play a part is nevertheless very useful.

2. The default behaviour. Text in brackets such as "(the laminated mahogany box)" is printed, on its own line.

3. Examples. (a) In the following, asking to TAKE TOWER results in the parser choosing the souvenir model (because of the "does the player mean..." rule making the alternative unlikely), and then explaining itself by saying "(The little one, obviously.)" instead of "(the souvenir model Eiffel Tower)".

```
The Champs du Mars is a room. The great Eiffel Tower is here. "The
great Tower stands high over you." The souvenir model Eiffel Tower
is here. "Comparatively tiny is the souvenir version." The great Eiffel
Tower is fixed in place. Does the player mean taking the great Eiffel
Tower: it is very unlikely.


Rule for clarifying the parser's choice of the model tower: say "(The
little one, obviously.)"
```

4. A note about actions. This activity takes place during the process of understanding the player's command, when the action that will take place is not fully known. So if the player types "TAKE SHOEBOX", this activity would happen when SHOEBOX is being examined for meaning. Inform knows that the action will be taking, but nothing else. That means attaching a proviso like "... while taking a container" to a rule for this activity will cause the rule to have no effect - whereas "... while taking" would be fine.

Start of Chapter 18: Activities

Back to §18.29. Deciding the scope of something

Onward to §18.31. Asking which do you mean

§18.31. Asking which do you mean

1. When it happens. When the player has typed an ambiguous noun reference, and Inform has not been able to decide what was meant.

2. The default behaviour. A question such as "Which do you mean, the laminated mahogany box or the boom box?" is printed. (This activity shapes the question: it is not responsible for parsing the answer. It would be very mysterious to write rules for this activity such that nothing is printed, because the player would then have no idea what to type.)

3. Examples. The question is harder to print than may first appear, since one must not simply list the options, but also take into account collections of plural objects ("Which do you mean, the gold-tipped pen or a gold coin?"). It is probably better not to try to rewrite this.

(a) But we can place notes before or after: here is a verbose explanation for beginners to IF.

```
Before asking which do you mean: say "Okay, so I'm going to have to
ask a question now: you've typed something ambiguous, and I don't
know which noun you're referring to."

After asking which do you mean: say "(Just type a word or two to give
me more information.)"
```

(b) We can also use this activity as a context for other activities. For instance:

```
The Champs du Mars is a room. The great Eiffel Tower is here. "The
great Tower stands high over you." The souvenir model Eiffel Tower
is here. "Comparatively tiny is the souvenir version." The great Eiffel
Tower is fixed in place. Understand "actual" as the great Tower.

Rule for printing the name of the great Tower while asking which do
you mean: say "actual Tower". Rule for printing the name of the
souvenir tower while asking which do you mean: say "souvenir".
```

causes TAKE TOWER (for instance) to produce a nice tidy question in reply: "Which do you mean, the actual Tower or the souvenir?"

4. A note about actions. This activity takes place during the process of understanding the player's command, when the action that will take place is not fully known. So if the player types "TAKE SHOEBOX", this activity would happen when SHOEBOX is being examined for meaning. Inform knows that the action will be taking, but nothing else. That means attaching a proviso like "... while taking a container" to a rule for this activity will cause the rule to have no effect - whereas "... while taking" would be fine.

Start of Chapter 18: Activities

Back to §18.30. Clarifying the parser's choice of something

Onward to §18.32. Supplying a missing noun/second noun

Example 368: **Apples** Prompting the player on how to disambiguate otherwise similar

objects.

Example 369: **Originals** Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

Example 370: **Walls and Noses** Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

---

§18.32. Supplying a missing noun/second noun

1. When it happens. (Two different activities here, but identical except for applying to different nouns.) This happens when an Understand sentence fails to supply a noun for an action which requires one. For example, in the sentence 'Understand "seize" as taking.' - the "taking" action is incompletely specified, because it requires a noun, and there's no noun in the command to be understood.

Note that this is not what happens if it's the player who fails to supply the noun. That is, suppose the player types a half-finished command like TAKE, which can't be matched against (for example) 'Understand "take [things]" as taking.' because the player didn't name any thing(s). Typically a story will reply to such a command with a question asking for clarification, but sometimes it makes guesses about what was meant. The "supplying a missing noun" activity plays no part in this guesswork, and can't influence it: that's the task of the "does the player mean" rulebook.

Suppose we do have the first of these cases, then. "Supplying a missing noun" takes place to remedy the problem. It can either:

(i) Set a noun, printing text like "(presumably the black bag)" if it wants, in which case the action goes forward, though it is still subject to the full rules on accessibility exactly as any other action would be; or

(ii) Make no choice, in which case no action takes place and the player's command is rejected. If the activity printed nothing, Inform will produce a generic reply to the player that "You must supply a noun.".

2. The default behaviour. In the default grammar for Inform, only three such half-finished actions are ever Understood. One is "going" with no direction, for which this activity simply prints a refusal. The other two are the two undirected senses, "smelling" and "listening". In each case, the "supplying a missing noun" activity sets the noun to the current location: so, for instance, typing the bare command "listen" might generate the action "listening to the Shoreline".

3. Examples. (a) This is the definition Inform uses to make "listen" work as outlined above:

```
Rule for supplying a missing noun while listening (this is the ambient
sound rule):
    now the noun is the location.
```

(b) It can be elegant to allow second nouns to be dropped with habitual actions, or where the choice is obvious:

```
Understand "unlock [something]" as unlocking it with.
```

```
Rule for supplying a missing second noun while unlocking:
    if the skeleton key is carried, now the second noun is the skeleton
key;
    otherwise say "You will have to specify what to unlock [the noun]
with."
```

Note that, in order for our activity to succeed, we do need to supply a grammar line allowing the player to try "unlocking it with" using only one noun. Otherwise, the command "unlock something" will still produce the question "What do you want to unlock the door with?"

Start of Chapter 18: Activities

Back to §18.31. Asking which do you mean

Onward to §18.33. Reading a command

Example 371: ★ **Latin**

Lessons Supplying missing nouns and second nouns for other characters besides the player.

Example 372: ★ **Minimal Movement** Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

---

§18.33. Reading a command

1. When it happens. When reading a command from the keyboard.

2. The default behaviour. Print the prompt text; wait for the player to type something and press return. Reject an entirely blank line, and treat a command beginning "oops" as a correction to the previous one. This is a fairly complicated business, so it is probably best not to change the "for" rules for this activity: "before", and especially "after", are another matter. (Note, however, that if Inform does reject a blank line and ask for another then this all happens inside the "for" rules: no "after" occurs after the blank line, nor does a "before" happen before the second attempt by the player. It is all a single round of the activity, not two.)

3. Examples. (a) To lead absolute beginners in gently:

```
Before reading a command while the turn count is 1, say "(This is
your chance to say what the protagonist should do next. After the '>',
try typing 'take inventory'.)"
```

(b) The following responds politely but firmly if the player tries to type "please look", say, instead of just "look":

```
After reading a command:
    if the player's command includes "please":
        say "Please do not say please.";
        reject the player's command.
```

To explain. Fragments of what the player has typed are called snippets: "the player's command" is the entire thing. We can test if a snippet matches a given pattern like so:

if (snippet) **matches** (topic)**:**

This condition is true if the given snippet exactly matches the specification. Example:

```
if the player's command matches "room [number]", ...
```

will be true if the command is ROOM 101, but not if it's EXPLORE ROOM 7.

if (snippet) **does not match** (topic)**:**

This condition is true if the given snippet does not exactly match the specification.

if (snippet) **includes** (topic)**:**

This condition is true if the given snippet includes words matching the specification, either at the beginning, in the middle, or at the end. Example:

```
if the player's command includes "room [number]", ...
```

will be true if the command is ROOM 101, EXPLORE ROOM 7, or ROOM 22 AHOY, but not if it's VISIT ROOM GAMMA 7.

if (snippet) **does not include** (topic)**:**

This condition is true if the given snippet does not include any run of words which matches the specification.

Lastly, we took drastic action with another new phrase:

reject the player's command

This phrase should be used only in rules for the "reading a command" activity. It tells Inform not to bother analysing the text further, but to go back

to the keyboard. (No time passes; no turn elapses; nothing happens in the simulated world.)

(c) An improved version takes commands like "please drop the coin" and strips "please" from them, but then allows them to proceed normally:

```
After reading a command:
    if the player's command includes "please":
        say "(Quelle politesse! But no need to say please.)";
        cut the matched text.
```

"Matched text" is a snippet containing the words which matched against the pattern in the most recent "includes" condition, so in this case it contains just the single word "please". Two phrases allow snippets to be altered:

replace (snippet) **with** (text)

This phrase should be used only in "after" rules for the "reading a command" activity; it replaces the snippet of command, usually the "matched text" found immediately before, with the given text. Example:

```
    if the player's command includes "room [number]":
        replace the matched text with "office".
```

cut (snippet)

This phrase should be used only in "after" rules for the "reading a command" activity; it removes the snippet of command. Example:

```
    if the player's command includes "or else":
        cut the matched text.
```

Note that "replace" and "cut" can only be used in "after reading a command" rules: not when an action has been chosen and has gone ahead into its rulebooks. Once the "reading a command" activity has finished, the command is final.

(d) To make the word "grab" an abbreviation for "take all":

```
After reading a command:
    if the player's command matches "grab", replace the player's
command with "take all".
```

("Snippet" is actually a kind of value, so we could say "Ah, you typed '[the player's command]'!" or some such if we liked. But in practice only three snippets are likely to be useful: the two mentioned above, "player's command" and "matched text", and the "topic understood", used when matching the "[text]" token in command grammar.)

(e) Finally, we can make still more detailed alterations to the text of the command using the techniques presented in the Advanced Text chapter. For instance:

change the text of the player's command to (text)

This phrase should be used only in "after" rules for the "reading a command" activity; it replaces the current command text entirely. Example:

```
After reading a command:
    let T be "[the player's command]";
    replace the regular expression "\p" in T with "";
    change the text of the player's command to T.
```

This converts the player's command to text, which is then manipulated by searching for any punctuation mark and replacing it with blank text (that is, deleted), and then put back again as the new command.

Start of Chapter 18: Activities

Back to §18.32. Supplying a missing noun/second noun

Onward to §18.34. Implicitly taking something

Example 373: ★★ **Cloves** Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

Example 374: ★★ **Fragment of**

a Greek Tragedy Responding to the player's input based on keywords only, and overriding the original parser entirely.

Example 375: ★★ **North by Northwest** Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed words.

Example 376: ★★★ **Complimentary Peanuts** A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

---

§18.34. Implicitly taking something

1. When it happens. When an action is tried which requires the actor (normally the player, of course) to be carrying something, but which is not in fact carried by that person. For instance, if the player types WEAR OVERCOAT in reference to a Moroccan overcoat currently draped over a chair.

2. The default behaviour. To print text such as "(first taking the Moroccan overcoat)" and then silently try taking the object in question (the overcoat). If the take succeeds, the silence means that nothing else is printed: if it fails, it will say why.

No matter what rules are written for this activity, it is impossible to use it to allow the action to go ahead even without the item. The activity allows us to change how, or if, an implicit take will happen, but not to change the consequences of failure. (To do that, we would need to say that "The ignore the carrying requirements rule does nothing", but this kind of unstitching of the action machinery needs to be done with caution.)

3. Examples. (a) Forbidding implicit takes for certain dangerous items. (This seems

especially fair if taking such items might cause death: the player will not wish to be killed on the strength only of our guess as to what he might be intending to do.)

```
Rule for implicitly taking the curare:
    say "Ordinarily you'd pick up the curare in order to be able to do
that, but this seems like a good moment for caution." instead.
```

(b) Changing the way the implicit action is reported for the player:

```
Rule for implicitly taking something (called target):
    try silently taking the target;
    if the player carries the target, say "You appropriate [the target]
first, of course. [run paragraph on]"
```

(c) Combining implicit takes when the noun and second noun must both be carried:

```
Rule for implicitly taking the noun when the second noun is a thing
and the second noun is not carried by the player:
    try silently taking the noun;
    try silently taking the second noun;
    say "(first taking both [the noun] and [the second noun])[line
break]"
```

(d) Making another character reply amusingly:

```
Rule for implicitly taking something which is carried by the player
when the person asked is Clark:
    say "'I don't see how I'm supposed to do that when you're holding
[the noun],' remarks Clark sulkily." instead.
```

(e) Causing implicit takes which wouldn't otherwise happen. Suppose we have a photographing action, and there are very small flowers which can't conveniently be snapped without being first picked. We then want an implicit take to occur, even though we wouldn't want this for other sorts of photography. So:

```
Check an actor photographing a flower:
    if the actor is not carrying the noun:
        carry out the implicitly taking activity with the noun;
        if the actor is not carrying the noun, stop the action.
```

Note that if the activity doesn't succeed in taking the item, it's expected to print some text explaining this, which is why we don't need to say anything further.

Start of Chapter 18: Activities

Back to §18.33. Reading a command

Onward to §18.35. Printing a parser error

Example 377: ★ **The Big Sainsbury's** Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

Example 378: ★ **Pizza Prince** Providing a pizza buffet from which the player can take as many pieces as he wants.

Example 379: ★★★ **Lollipop Guild** Overriding the rules to allow the player to show something to another character without first taking it.

---

§18.35. Printing a parser error

1. When it happens. The parser is the part of the run-time software, included in all works produced by Inform, which tries to match the player's command against the grammar provided by the work. When it is unable to make a valid match, the parser prints an error to the player: for instance,

```
> BIFURCATE TREE
That's not a verb I recognise.
```

There are more than twenty possible messages. The one which the parser wants to say is stored in the variable "latest parser error", which has the convenient kind "command parser error". This has the following possible values:

```
didn't understand error
only understood as far as error
didn't understand that number error
can't see any such thing error
said too little error
aren't holding that error
can't use multiple objects error
can only use multiple objects error
not sure what it refers to error
excepted something not included error
can only do that to something animate error
not a verb I recognise error
not something you need to refer to error
can't see it at the moment error
didn't understand the way that finished error
not enough of those available error
nothing to do error
referred to a determination of scope error
I beg your pardon error
noun did not make sense in that context error
can't again the addressee error
comma can't begin error
can't see whom to talk to error
can't talk to inanimate things error
didn't understand addressee's last name error
```

2. The default behaviour. Prints the message in question.

3. Examples. (a) Perhaps for newcomers:

```
After printing a parser error:
    say "If you are new to interactive fiction, you may like to try typing
HELP."
```

(b) Or to give the parser a certain amount of character:

```
Rule for printing a parser error when the latest parser error is the I
beg your pardon error:
    say "What's that? Speak up, speak up." instead.


Rule for printing a parser error:
    say "That's a rum thing to say, and no mistake." instead.
```

(c) This can be helpful for seeing what's going on:

```
Rule for printing a parser error:
    say "The [latest parser error] happened.";
    continue the activity.
```

Start of Chapter 18: Activities

Back to §18.34. Implicitly taking something

Onward to §18.36. Deciding whether all includes

Example 380: ★ **WXPQ** Creating a more sensible parser error than "that noun did not make sense in this context".

Example 381: ★★★ **Xot** Storing an invalid command to be repeated as text later in the game.

---

§18.36. Deciding whether all includes

1. When it happens. When parsing a command such as "take all", where the player uses "all" to signify everything within reach.

2. The default behaviour. The actual method used is complicated, as "all" is not as simple as it seems - "take all" would not include the player's own body, for instance, or the crescent moon. The point of this activity is to allow the normal method to be changed for given objects, or given kinds of object.

3. Examples. (a) Removing scenery from "all" (but see (4) below):

```
Rule for deciding whether all includes scenery: it does not.
```

The phrases "it does" and "it does not" make a decision.

(b) Ensuring that a given thing, which might otherwise be excluded, is included:

```
Rule for deciding whether all includes the oval roof: it does.
```

4. The Standard Rules already uses this. Note that the Standard Rules already stocks this activity with several rules:

```
exclude scenery from take all rule
exclude people from take all rule
exclude fixed in place things from take all rule
```

5. A note about actions. This activity takes place during the process of understanding the player's command, when the action that will take place is not fully known. So if the player types "TAKE SHOEBOX", this activity would happen when SHOEBOX is being examined for meaning. Inform knows that the action will be taking, but nothing else. That means attaching a proviso like "... while taking a container" to a rule for this activity will cause the rule to have no effect - whereas "... while taking" would be fine.

Start of Chapter 18: Activities

Back to §18.35. Printing a parser error

Onward to §18.37. Printing the banner text

§18.37. Printing the banner text

1. When it happens. The banner is the bibliographic masthead text, which typically looks something like this:

```
Relations
An Interactive Fiction by Emily Short
Release 1 / Serial number 050630 / Inform 7 build 2U98 (I6/v6.30 lib
6/10N) SD
```

(The serial and build numbers are those applying when the story file was last made: these ones are from the mid-2000s.) The banner is printed at the start of play, and when the player types "version" at the command line, and when say "[banner text]" occurs.

say "[the/-- banner text]"

This text substitution expands to the banner text giving bibliographic details

of the current story file, rather like the opening credits of a movie, or the title page of a book.

2. The default behaviour. Prints the text above, giving the title, the headline, the author, the release number, the date of compilation (that's the serial number: YYMMDD), and version numbers of the Inform components used to put the story together.

3. Examples. (a) Adding a line to the banner:

```
After printing the banner text, say "DRM authentication code:
13S-451-2034u75y65u%%a1248."
```

(b) Simplifying the banner:

```
Rule for printing the banner text: say "Welcome." instead.
```

Start of Chapter 18: Activities
Back to §18.36. Deciding whether all includes
Onward to §18.38. Printing the player's obituary
Example 382: ★ **Bikini Atoll** Delaying the banner for later.

§18.38. Printing the player's obituary

1. When it happens. The obituary is the text "*** You have died ***" or similar, sometimes followed by a final score, if the appropriate use option ("Use scoring.") is in force.

2. The default behaviour. Printing the aforementioned text, then the final score, and reducing the status line to a largely blank state.

3. Examples. Here's one way to add to the verdict of history:

```
After printing the player's obituary: say "And you visited [number of
visited rooms] place[s]."
```

Start of Chapter 18: Activities

Back to §18.37. Printing the banner text

Onward to §18.39. Amusing a victorious player

Example 383: ★ **Battle of Ridgefield** Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

Example 384: ★ **Finality** Not mentioning UNDO in the final set of options.

Example 385: ★ **Jamaica 1688** Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

§18.39. Amusing a victorious player

1. When it happens. When the player chooses "AMUSING" from the short menu of choices after a story has been won. Traditionally, this is where the author gets to point out quirky by-ways of the story, or make some final acknowledgements, or simply salute the player's perseverance. Note that the AMUSING option is only offered when the story has ended finally, and that it is only offered if there is at least one rule present in the "for amusing a victorious player" rulebook.

2. The default behaviour. None. The "for amusing a victorious player" rulebook is empty by default, and no amusement is available.

3. Examples. The format would be like so:

```
Rule for amusing a victorious player: say "Hmm. You're easily
amused."
```

Start of Chapter 18: Activities

Back to §18.38. Printing the player's obituary
Onward to §18.40. Starting the virtual machine

Example 386: ★★ **Xerxes**
Offering the player a menu of things to read after winning the game.

---

§18.40. Starting the virtual machine

1. When it happens. This activity is provided solely as a "hook" for any low-level tasks which need to be performed when the virtual computer which runs Inform story files is starting up. This happens much earlier than "when play begins" rules, and should be used only as a last resort.

It should be remembered that Inform can produce story files for several different virtual computers. On some of these, it will not be safe to print any text during this activity, as the windows which would display such text do not yet exist.

2. The default behaviour. None.

3. Examples. No detailed examples will be given here, but the activity might be used (for instance) to set styles for the Glulx windows shortly to be brought into existence.

Start of Chapter 18: Activities
Back to §18.39. Amusing a victorious player
Onward to Chapter 19: Rulebooks: §19.1. On rules

Example 387: ★ **Blankness**
Emptying the status line during the first screen of the game.

Examples from Chapter 18: Activities

Start of this chapter

334 **Example** **Ant-Sensitive Sunglasses** *RB*

What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

Suppose we want to create an object -- or maybe even a series of objects -- that warp the player's perception of every room description and object around him.

We've already seen some ways to create variations in text. For instance, we could make a room description with if substitutions in it, like so:

```
The Kitchen is a room. "[if the player is wearing the
sunglasses]Are ants coming out of the sink? No,
probably no.[otherwise]A small kitchen tucked into a
corner of the vacation house. There is storage space for
five or six cups, a sink, a two-ring stove; nothing else to
speak of.[end if]"
```

That works fine if we have one or two variations we want to add; it's not so good if we're going to have several items that work like the sunglasses, or if we want the sunglasses to override the description of every room in the house.

A slightly more flexible method is to use a substitution that calls out to a say phrase, like this:

```
The Kitchen is a room. "[kitchen-description]"

To say kitchen-description:
    if the player is wearing the sunglasses:
        say "Are ants coming out of the sink? No, probably
no.";
    otherwise:
        say "A small kitchen tucked into a corner of the
vacation house. There is storage space for five or six
cups, a sink, a two-ring stove; nothing else to speak of."
```

But again this doesn't handle the case of overriding multiple rooms at once very well.

When we reach a point where we need a given piece of text to be very flexible depending on the world model, it's time to use an activity.

Activities offer several advantages. One, we can create an activity like this:

```
Printing the room-description of something is an activity.
```

and then write a rule that applies to multiple rooms at once, like:

```
Rule for printing the room-description of a room when
the player wears the sunglasses:
    say "The walls look like they're covered with ants.
Just a coincidence, I'm sure."
```

Inform's usual rule-ranking also means that more-specific rules will override less-specific ones, so we could add

```
    Rule for printing the room-description of the Kitchen
when the player wears the sunglasses:
    say "Are ants coming out of the sink? No, probably
not."
```

and have that rule override the behavior of the activity just in the kitchen. Meanwhile, our base room descriptions remain straightforward and uncluttered by if-statements.

Several other examples will show how to hook activities into existing actions: Crusoe goes into detail about how how to make the descriptions of things more variable, and Aftershock demonstrates activities for describing the behavior of switchable devices.

Here, we preview all of those methods, just to get a sense of how they work and why they might be useful in controlling a game. Subsequent chapters go into more detail about the syntax of creating activities and the list of activities that are already defined by Inform.

```
"Ant-Sensitive Sunglasses"
```

Part 1 - Procedure

To add a new activity to an existing Inform rule, we need to do three things:

1) Define our new activity.

2) Give a basic rule that says what is supposed to happen when that activity occurs, as in "Rule for..."

3) Replace the existing rule in Inform's rulebooks with a new one that calls on our activity.

Here we do this with examining:

Section 1 - Item Description

Printing the description of something is an activity.

Now, by default, we want to print the description property; we just want the option to write some extra rules overriding that property. So we tell Inform that our most basic rule for printing the description of something is just to give that description text:

```
Rule for printing the description of something (called
item):
    if the description of the item is not "":
        say "[description of item][paragraph break]";
    otherwise:
        say "You see nothing special about [the item].".
```

Next, we need the standard examining rule to look at our printing-the-description activity:

The activity-based examining rule is listed instead of the standard examining rule in the carry out examining rules.

```
This is the activity-based examining rule:
    carry out the printing the description activity with the
noun;
    rule succeeds.
```

Now we do the same thing to room descriptions.

```
Section 2 - Room Description
```

```
Printing the room-description of something is an activity.
```

```
Rule for printing the room-description of a room (called
item):
    if the description of the item is not "":
        say "[description of item][paragraph break]";
    otherwise:
        do nothing instead.
```

```
The activity-based room description body text rule is
listed instead of the room description body text rule in
the carry out looking rules.
```

Our replacement rule this time around is a little bit trickier just because the rule that we're replacing is a complicated one: describing a room already checks to see whether there's light to see by, whether the player has turned off room descriptions when he enters a room for the second time, and whether the player character is (say) inside a closed box he can't see out of.

But all of those details are re-copied from the standard rules, and the important thing is that, at the end, we again carry out our activity.

```
This is the activity-based room description body text
rule:
    if the visibility level count is 0:
        if set to abbreviated room descriptions, continue
the action;
        if set to sometimes abbreviated room descriptions
and
```

```
            abbreviated form allowed is true and
            darkness witnessed is true,
            continue the action;
        begin the printing the description of a dark room
activity;
        if handling the printing the description of a dark
room activity,
            say "It is pitch dark, and you can't see a thing.";
        end the printing the description of a dark room
activity;
    otherwise if the visibility ceiling is the location:
        if set to abbreviated room descriptions, continue
the action;
        if set to sometimes abbreviated room descriptions
and abbreviated form
            allowed is true and the location is visited,
continue the action;
        carry out the printing the room-description activity
with the location.
```

Section 3 - Device Description

```
Showing action of something is an activity.


Rule for showing action of something (called item):
    if the item is switched on, say "[The item] is switched
on.";
    otherwise say "[The item] is switched off."


The activity-based described devices rule is listed
instead of the examine devices rule in the carry out
examining rules.


This is the activity-based described devices rule:
    if the noun is a device:
        carry out the showing action activity with the noun;
        now examine text printed is true.
```

Report switching on something:
    say "You flip a switch. ";
    carry out the showing action activity with the noun instead.

Part 2 - Scenario

The Kitchen is a room. "A small kitchen tucked into a corner of the vacation house. There is storage space for five or six cups, a sink, a two-ring stove; nothing else to speak of."

The microwave is a device in the Kitchen.

South of the Kitchen is the Living Area. The description of the Living area is "A whitewashed living/dining/reclining area in what used to be a shepherd's stone hut, but now costs vacationers 600 euros a week. It offers no mod cons, only a straight view of the Mediterranean and a wobbly writing table."

Rule for printing the room-description of a room when the player wears the sunglasses:
    say "The walls look like they're covered with ants. Just a coincidence, I'm sure[antsy]."

Rule for printing the room-description of the Kitchen when the player wears the sunglasses:
    say "Are ants coming out of the sink? No, probably not[antsy]."

Rule for printing the description of something (called the item) when the player wears the sunglasses:
    say "[The item] [are] [one of]ant-colored[or]ant-

```
legged[or]covered in ants[at random][antsy]."


Rule for showing action of the microwave:
    say "The microwave hums meaningfully to itself."


Rule for showing action of the microwave when the
player wears the sunglasses:
    say "The microwave hums as though inhabited by a
billion ants[antsy]."


The player carries sunglasses of freakiness and an
apple. The apple is edible. The sunglasses are
wearable.


ant-paranoia is a number that varies.


To say antsy:
    increase ant-paranoia by 1;


Every turn:
    if the ant-paranoia is greater than 3:
        say "Augh! AUUUGH! GET THEM OFF--";
        end the story saying "You have lost your mind."


Test me with "look / turn on microwave / turn off
microwave / x apple / x sunglasses / s / wear
sunglasses / look / x apple / n / turn on microwave".
```

ExampleAARP-Gnosis
An Encyclopedia set which treats volumes in the same place as a single object, but can

Suppose we have a complete Encyclopedia in our game. The player is allowed to pick up the whole set (there must not be too many volumes), but also to do things with individual volumes, and indeed to scatter these volumes all over the place. Putting a volume back in the same place as the rest of the Encyclopedia should, however, restore it to the collective. We will start out by defining general rules for collectives like this:

```
"AARP-Gnosis"

Fitting relates various things to one thing (called the
home). The verb to fit means the fitting relation.
Definition: a thing is missing if it is not part of the home
of it.

A collective is a kind of thing.

Before doing something to something which is part of a
collective:
    let space be the holder of the home of the noun;
    move the noun to the space.

Instead of examining a collective:
    say "[The noun] consists of [the list of things which
are part of the noun]."
```

Now the real work begins. One reason to make this an activity is that we might easily want to override it for specific objects; for instance, the generic collecting activity here would not deal properly with collectives of clothing where some items might be worn and others not. In that case, we would want to write another, more specific "collecting" activity to handle the complexities of fashion.

```
Collecting something is an activity.


Every turn:
    repeat with item running through collectives:
        carry out the collecting activity with the item.


To remove (item - a thing) when empty:
    let space be the holder of the item;
    if the number of things which are part of the item is 0:
        now the item is nowhere;
    if the number of things which are part of the item is 1:
        let the last thing be a random thing which is part of
the item;
        move the last thing to the space;
        now the item is nowhere.


Before collecting a thing (called the item):
    remove item when empty;
    let space be the holder of the item;
    if space is not a thing and space is not a room:
        if something (called the other space) contains at
least two things which fit the item, move item to the
other space;
        if a room (called the other space) contains at least
two things which fit the item, move item to the other
space;
        if someone (called the owner) carries at least two
things which fit the item, move item to the owner.


Rule for collecting a thing (called the item):
    let space be the holder of the item;
    if space is a thing or space is a room:
        repeat with component running through things held
by the space:
            if the component fits the item, now the
component is part of the item;
        remove item when empty.
```

And now for a cheerful scenario:

The Boise Memorial Library is a room. "A concrete box of a room, roughly eight feet by fourteen, which contains all the fallout shelter has to offer by way of entertainment. Someone with a grim sense of humor has tacked a READ! literacy poster to the door, as though there were anything else to do while you await the calming of the Geiger counters." The shelf is a supporter in the Library. "A battered utility shelf stands against the south wall."

The New Idahoan Encyclopedia Set is a collective. Volume A-Aalto fits the Encyclopedia. It is part of the Set. Volume AAM-Aardvark fits the Encyclopedia. It is part of the Set. Volume Aarhus-Aaron fits the Encyclopedia. It is part of the Set. Volume AARP-Gnosis fits the Encyclopedia. It is part of the Set. Volume Gnu-Zygote fits the Encyclopedia. It is part of the Set. The Set is on the shelf.

Let's have the Encyclopedia describe itself differently depending on whether it's all in one place:

After printing the name of the Set when something missing fits the Set:
    say " (missing [a list of missing things which fit the Set])"

Before printing the name of the Set when the number of missing things which fit the set is 0:
    say "complete ".

Test me with "get aarhus-aaron / look / inventory / get aam-aardvark / look / get gnu-zygote / look / get aarp-gnosis / look / inventory / drop set / look / get set / get a-aalto / inventory".

ExampleAftershock

Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times. *RB*

The built-in behavior of Inform is to print a line after a device is examined, saying whether the item is on or off. This is often inappropriate, and we could simply turn off that behavior in general by instructing Inform to ignore the "examine described devices rule" (see the chapter on rulebooks).

Perhaps, though, we would like continue to have a short passage about the action of any switched on device; we'd just like a little more control over what it says from time to time. And in that case, we might change the rule to give a new activity control over that portion of the description:

```
"Aftershock"

Section 1 - Showing actions

Showing action of something is an activity.

Rule for showing action of something (called item):
    if the item is switched on, say "[The item] is switched
on.";
    otherwise say "[The item] is switched off."
```

Borrowing from the rulebooks chapter, we can replace the standard "examine described devices" rule with something that uses this activity.

```
The new described devices rule is listed instead of the
examine devices rule in the carry out examining rules.

This is the new described devices rule:
    if the noun is a device:
        carry out the showing action activity with the noun;
```

```
now examine text printed is true.
```

Thus far we have essentially replicated the original behavior, but we've made it possible to write specialized behavior for devices, and to invoke that behavior in other places:

```
Report switching on something:
    say "You flip a switch. ";
    carry out the showing action activity with the noun
instead.
```

This might be useful for an electric lamp kind:

```
Section 2 - Electric Lamps

An electric lamp is a kind of device.

Rule for showing action of an electric lamp (called item):
    if the item is switched on, say "[The item] is lit[if the
number of visible lit things is greater than 1], competing
with [the list of visible lit things which are not the item]
[end if].";
    otherwise say "[The item] is dark."

Carry out switching on an electric lamp: now the noun is
lit. Carry out switching off an electric lamp: now the
noun is unlit.

Section 2 - The Scenario

The time of day is 3:47 AM. When play begins, now the
right hand status line is "[time of day]".
```

```
The Downstairs Hallway is a dark room. "The only room
in the house with no furniture and almost nothing on the
walls. At times like this you always notice the crack in
the plaster, originating near the light fixture and running
almost all the way to the wall."


A plastic jug of filtered water is in the Downstairs
Hallway. The description is "Five gallons, not that that
will last you very long, hot as it has been lately."


The crack is scenery in the Hallway. The description is
"No, the ceiling isn't going to fall on you today."


The light fixture is an electric lamp in the Hallway. It is
switched on, lit, and scenery. The description is "A plain
globe of frosted glass containing the light bulb. Nothing
special, and you never think about it except when, as
now, you are forced to spend hours in this room."


The flashlight is an electric lamp carried by the player.
The description is "A shiny red flashlight." The portable
radio is a device carried by the player. The description is
"A small battery-operated radio which you received for
free with your subscription to US News & World Report.
It has served you well through many earthquakes past."
```

And with our activity, we can override the flashlight's electric lamp behavior with new behavior:

```
Rule for showing action of the flashlight:
    if the flashlight is switched on, say "A strong, narrow
beam of light shines from the flashlight.";
    otherwise say "It is currently switched off."
```

...or give special actions for the radio:

```
Rule for showing action of the radio:
```

```
    if the radio is switched on, say "Through the static,
you pick up pieces of discussion: a 6.7 on the Richter
scale, epicenter... something about Topanga... but it
crackles out again.";
    otherwise say "The radio is silent. You're saving the
batteries."



Instead of listening in the presence of the switched on
radio:
    carry out the showing action activity with the radio
instead.



Test me with "examine light / switch light off / switch
flashlight on / switch radio on / examine radio / examine
flashlight".
```

ExampleCrusoe

337 *RB*

Adding a "printing the description of something" activity.

Suppose we want to add rules so that any time we examine a charred object (and most of our objects can be charred), a line about the charring is appended to the end of the object description. We could use "after examining...", but perhaps we would prefer for the sentence about the charring not to appear in its own paragraph.

This is an ideal occasion for a new activity. We look at the action index for "examining" to identify the rule that causes the old behavior (in this case, the "standard examining rule"); replace this with a new rule that calls our activity; and write our "printing the description" activity in such a way that it uses an object's description without forcing a paragraph return afterward.

Then we will use "after printing the description" to add our line about charring, and make sure that the paragraph return does occur before the prompt.

So:

```
"Crusoe"
```

```
Section 1 - Creating our New Activity
```

```
The fancy examining rule is listed instead of the
standard examining rule in the carry out examining
rules.
```

This instruction replaces a normal piece of the examine action, the standard examining rule, with another one of our own devising. (The replacement of the standard examining rule will be explained in more detail in the chapter on rulebooks.)

```
Printing the description of something is an activity.
```

```
This is the fancy examining rule:
    carry out the printing the description activity with the
noun;
    rule succeeds.
```

All we have done here is enclose what is usually just a rule inside an activity. This means that we can now write before and after rules for the activity, and also add special instructions like "Rule for printing the name of something while printing the description of something" -- this may not be likely to arise often, but Inform now has the concept of "printing the description of something" as a separate context of action. Next we add the modification that lets us append to the description without a new line:

```
Rule for printing the description of something (called
item):
    if the description of the item is not "":
        say "[description of item] [run paragraph on]";
    otherwise:
        say "You see nothing special about [the item]. [run
paragraph on]".
```

"run paragraph on" here will mean that we do not get a paragraph break following the description, even if it ends with a period. We also insert a

space, so that our follow-on comments will be properly punctuated.

```
After printing the description of something charred:
    say "It is charred." instead.
```

The instead at the end of this line stops Inform for going on with any other "after printing the description of..." rules.

The standard library also has rules for printing additional text about containers and supporters with visible contents, and devices that are switched on; with this current system, we could add those as "after printing the description" rules as well, building up a complete paragraph if we wanted. But for simplicity we won't exemplify all of that here. The effects would be much the same as with the "charred" line.

Now, because we want to make sure that we always do get a paragraph break after our description, we add this rule last after all the other rules. "Last" and "first" rules are covered in more detail in the chapter on rulebooks.

```
Last after printing the description of something:
    say paragraph break.
```

```
Section 2 - The Scenario
```

```
The Desert Isle is a room. "A pale expanse of sand,
here and there developing into hillocks of grass, and a
small clump of palms. The water is shallow here, and
there are other islands within swimming distance -- or
even wading distance, perhaps -- but none of them is
any larger than your island, so it doesn't seem worth the
trouble of visiting.
```

```
A few hundred feet out, the water turns darker blue, the
sea floor drops away, and there is nothing to be seen all
the way down to the horizon, except a couple of fluffy
clouds, and an occasional bird.
```

```
The remains of your fire smolder in the stone-lined pit."


A thing can be charred or whole. A thing is usually
whole. Instead of burning something: say "You hold [the
noun] to the fire until it flares and chars."; now the noun
is charred.


The player carries a stick. The description of the stick is
"A strip of palm from the woodiest part of the leaf, about
a foot and a half long."


The player carries a glass bottle and a piece of paper.
The description of the paper is "A single blank sheet." In
the glass bottle is a grain of sand. The glass bottle is
openable and open. Instead of burning the glass bottle:
say "You hold the bottle to the flame, but it grows
uncomfortably warm."


Instead of burning the grain of sand: say "You drop the
grain into the fire pit, where it becomes indistinguishable
from all the others."; now the grain of sand is nowhere.
Instead of dropping the grain of sand: now the grain of
sand is nowhere; say "You return the grain of sand to its
brethren."
```

The player's description is handled in an unusual way, and this will produce a space paragraph break there where it should not. Instead, therefore, we will add an instead for examining the player (probably a good idea anyway):

```
Instead of examining the player:
    say "You are sunburned and there is sand in cracks
you didn't know existed."
```

```
Test me with "i / x stick / x bottle / x sand / x paper / x
me / burn stick / x stick / burn paper / x paper".
```

The "printing a description" activity may be useful for other games, and can be imported just by lifting section 1.

---

338 **Example** **Hays Code** *RB*

Clark Gable in a pin-striped suit and a pink thong.

The following burlesque was considered too much for the tender readers of Chapter 3, since it involved explicit use of listing and persuasion:

```
"Hays Code"

The Movie Set is a room. Clark Gable is a man in the
Movie Set. "Clark leans on a polystyrene pillar, wearing
[a list of unconcealed things worn by Clark] with his
usual aplomb." Persuasion rule for asking Clark to try
doing something: persuasion succeeds.

Clark is wearing a pin-striped suit and a pink thong.
Rule for deciding the concealed possessions of Clark: if
the particular possession is the thong and Clark is
wearing the suit, yes; otherwise no.

Test me with "clark, remove suit / look / clark, remove
thong / look".
```

---

339 **Example** **Shipping Trunk** *RB*

A box of baking soda whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

First to lay some groundwork:

"Shipping Trunk"

A chest is a kind of container. A chest is always openable. A lid is a kind of supporter. A lid is part of every chest.

Before opening a chest when something (called the obstruction) is on a lid which is part of the noun:
    say "Better remove [the obstruction]." instead.

A thing can be innocent or smelly.

The Storage Unit is a room. The shipping trunk is a closed chest in the Storage Unit. The trunk contains some garlic, a loaf of moldy sourdough, a mildewy bathtowel, a pair of unwashed socks, two dead trout, and a box of baking powder. The garlic, trout, sourdough, bathtowel, and socks are smelly. The baking powder is innocent.

The shipping trunk's lid supports a small card. The description of the small card is "'Please, please do not open this trunk.'"

After opening the trunk:
    if the trunk had been open:
        say "You steel yourself...";
        continue the action;
    otherwise:
        say "There roils up from inside an indescribable funk, which, when you can see straight, you have no trouble attributing to the presence of [a list of smelly

```
things in the trunk]. You also note [a list of innocent
things in the trunk] in the corner.".
```

And now, with that preparation:

```
Before printing the name of the baking powder when the
powder is in a container which contains a smelly thing:
say "completely ineffective ".


Test me with "open trunk / examine card / get card /
open trunk / get powder / inventory".
```

Example **Trachypachidae Maturin 1803** *RB*

340

Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

```
"Trachypachidae Maturin 1803"


A bottle is a kind of container. Bottles are usually
openable, transparent, and closed. A cork is a kind of
thing. A cork is in every bottle.


Understand "cork [something]" as corking.


Understand the command "stopper" as "cork".


Understand "uncork [something]" as uncorking.
```

```
Corking is an action applying to one thing.

Check corking:
    if the noun is not a bottle, say "[The noun] cannot be
corked." instead.

Carry out corking:
    try closing the noun.

Uncorking is an action applying to one thing.

Check uncorking:
    if the noun is not a bottle, say "[The noun] cannot be
uncorked." instead.

Carry out uncorking:
    try opening the noun.

Understand "close [something] with [something
preferably held]" as corking it with.

Understand "cork [something] with [something
preferably held]" as corking it with.

Corking it with is an action applying to one thing and
one carried thing.
```

Check corking it with:
    if the noun is not a bottle, say "[The noun] cannot be corked." instead;
    if the second noun is not a cork, say "[The second noun] will not fit in [the noun]." instead.

Carry out corking it with:
    try inserting the second noun into the noun instead.

Instead of closing a bottle:
    if a cork (called the item) is carried by the player, try inserting the item into the noun instead;
    otherwise say "You need a stopper of some kind."

Instead of opening a bottle:
    if a cork (called the item) is in the noun, try taking the item instead;
    otherwise say "[The noun] has no stopper."

Carry out inserting a cork into a bottle:
    now the second noun is closed.

After inserting a cork into a bottle:
    say "You stopper [the second noun] with [the noun]."

Before taking a cork when the noun is in a closed bottle (called the item):
    now the item is open.

Instead of taking a cork when the noun is in a bottle (called the item):
    move the noun to the player;
    say "You pull [the noun] from [the item]." instead.

Before printing the name of a bottle (called target) while not inserting, taking, searching, or removing:
    if the target is closed, say "sealed ";
    otherwise say "now open ".

After printing the name of a bottle (called target) while not inserting, searching, examining, or removing:
    if the target contains a noncork thing, say " containing [a list of noncork things in the target]";
    omit contents in listing.

Instead of examining a bottle:
    say "[The noun] contains [a list of noncork things in the noun]."

Definition: a thing is noncork if it is not a cork.

The Doctor's Cabin is a room. "A dark, cramped triangle, like a slice of cake, except that its sharp end has been cut off: and so low that a moderately tall man would strike his head on the deck above if he were to stand upright. Every free surface is covered with sheets of best Venetian looking-glass, to increase the light filtering in. Long use and the carpenter's ingenuity have packed in a folding cot and table, and lockers are built into unlikely places: lockers filled with specimens, skeletons, sketches, drafts and serial letters." The jug is a bottle in the Doctor's Cabin. The jug contains a beetle. The description of the beetle is "The doctor assures you that it is a nondescript."

Test me with "get jug / x jug / open jug / x jug / i / x cork / cork jug / i / uncork jug / i / x jug / get beetle / i / close jug / i / x jug".

Example **Chronic Hinting Syndrome** *RB*

Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

Suppose we have a conversation system in which it is important to keep track of which subjects the player has heard mentioned. If we're careful to mark subjects in brackets, we can use the "printing the name of" activity to record which things have been mentioned so far:

```
"Chronic Hinting Syndrome"

A subject is a kind of thing.

Knowledge relates various people to various subjects.
The verb to know means the knowledge relation.

Awareness relates various people to various subjects.
The verb to be aware of means the awareness relation.

Definition: a subject is pending if the player is aware of it
and it is not known by the player.

Instead of thinking:
    if the number of pending subjects is 0, say "You have
no fresh leads at the moment.";
    otherwise say "You recall that thus far you have not
followed up with questions about [the list of pending
subjects]."

After printing the name of a subject (called idea):
    now the player is aware of the idea.
```

Now suppose that as an added convenience for the player, we let him turn on a mode in which useful conversation topics are always automatically highlighted in the text, so he doesn't waste his time trying to follow up dead leads:

```
Setting is a kind of value. The settings are bright and
dull. Understand "on" as bright. Understand "off" as dull.

Highlighting is a setting that varies. Highlighting is dull.

Understand "highlighting [setting]" as setting
highlighting. Setting highlighting is an action out of
world, applying to one setting.

Carry out setting highlighting:
    now highlighting is the setting understood.

Report setting highlighting:
    say "Highlighting is now [if highlighting is
dull]off[otherwise]on[end if]."

Before printing the name of a subject (called idea) when
highlighting is bright:
    unless the player knows the idea:
        say "[bold type]".

After printing the name of a subject when highlighting is
bright:
    say "[roman type]".
```

...And the rest is peripheral.

The Sickbay is a room. "A place arranged for Nathan's comfort, since his sickness has been prolonged and because he becomes so irritating when not comfortable." The Hallway is outside from the Sickbay.

A supporter can be untried or rejected. A supporter is usually untried.

The Sickbay contains a wobbly pedestal, a table, and a sickbed. Understand "bed" as the sickbed. The pedestal, the table, and the sickbed are supporters. Nathan is a man on the sickbed. The sickbed is scenery. The initial appearance of the wobbly pedestal is "A wobbly pedestal near the door has sometimes been known to support vases of flowers, but is currently bare." The initial appearance of the table is "There is also a table of a more ordinary sort."

Nathan can be active or passive.

After printing the name of Nathan: now Nathan is passive.

Instead of putting the sculpture on the table:
    now the table is rejected;
    say "'[Not there],' [Nathan] snaps. 'The table is way too far from the sickbed.'"

Instead of putting the sculpture on the sickbed:
    now the sickbed is rejected;
    say "'[Not there],' [Nathan] rebukes you. 'You don't want me knocking it over if I roll around. In pain.'"

Instead of putting the sculpture on the pedestal:

now the pedestal is rejected;
say "The pedestal starts to wobble so ominously that you don't dare let go.

'[Not there],' says [Nathan]. 'That thing is falling apart.'"

Before putting something on the down: try dropping the noun instead.

To say not there:
if all the supporters are rejected:
say "Look, the floor would be fine";
otherwise if the number of rejected supporters is 1:
say "Yeah, anywhere but there, I'm afraid";
otherwise:
say "Come on, use your head -- I can't be watching you all the time, I'm sick".

Instead of going outside when the player is carrying the sculpture:
say "You've still got this sculpture to get rid of."

Instead of going outside when the breakage is pending:
say "You can't very well smash in front of [Nathan] his prize sculpture and then just scamper off without saying something. Appealing though the thought is at the moment."

Instead of going outside when a subject which is not the breakage is pending:
say "'Yeah, go ahead,' says [Nathan], with a martyr-like air. 'It's probably best that you don't hear about [the random pending subject]. It's not something I'd go into normally.'"

The breakage is a subject. The description is "It's not a big deal. I'll just buy a new [mental wave generator].' A slight awkward pause. 'I mean, this one was a [gift], but don't worry about it". Understand "accident" or "smashing" or "breaking" or "shard" or "mishap" or "shards" or "mistake" as the breakage. Understand "sculpture" as the breakage when the player is not carrying the sculpture.

Instead of saying sorry in the presence of Nathan when the player is aware of the breakage:
    try asking Nathan about the matter of breakage.

Instead of asking Nathan to try saying sorry when the player is aware of the breakage:
    try asking Nathan about the matter of breakage.

The mental wave generator is a subject. The description is "They're kind of expensive but I can save up. I really need one, though, because of my [dreams]".

The dreams is a subject. The description is "They're not the kind of dream you want to have.' He closes his eyes and contemplates these undesirable dreams. 'Have you ever woken up convinced you were dead? No, probably not. Well... At any rate, I need the [generator]. Oh, don't worry, they're expensive but not so expensive that I won't be able to save up for another, in a few months".

The gift is a subject. The description is "[The mental wave generator] was a present from a girl named [Shari]. Actually I'm not sure she'd take to being called a girl".

Shari is a subject. The description is "Look, let's just not go into it, okay? I don't really want to relive all that right

now. I still have a six-inch [scar] shaped like a banana in the middle of my back".

The scar is a subject.

Instead of asking Nathan about the matter of the scar:
    say "Nathan clears his throat, lowers his voice, and begins to tell you the story...";
    end the story saying "End of Demo -- Register to Continue!!"

Understand "ask [someone] about [any subject]" as asking it about the matter of.

Asking it about the matter of is an action applying to one thing and one visible thing.

Check asking it about the matter of:
    if the player is not aware of the second noun, say "What [second noun]?" instead;
    if the noun does not know the second noun, say "'I've no idea,' replies [the noun]." instead;
    if the player knows the second noun, say "You've already covered that. The response was '[description of the second noun].'" instead.

Carry out asking it about the matter of:
    now the player knows the second noun.

Report asking it about the matter of:
    say "'[description of the second noun],' says [the noun]."

Instead of telling Nathan about something:
    say "He pinches the bridge of his nose. 'I can't really follow this right now,' he says. 'I'm sorry.'"

Instead of asking Nathan about something:
    say "He shrugs weakly."

When play begins:
    say "'Just put that down anywhere,' says [Nathan], as you come into the room. He's sitting in the sickbed with his legs straight out in front of him. 'I don't care where.' His voice is weak, but it sharpens up for the last remark: 'And don't make a lot of noise about it.'

Considering that he woke you from a sound slumber to beg you to bring this thing over, his attitude is a bit much. You stare dubiously at the awkward crystal sculpture in your hands.";
    now Nathan knows every subject.

Instead of asking Nathan about something while the player carries the sculpture, say "[Nathan] moans dramatically and refuses to be drawn into conversation."

The player is carrying an awkward crystal sculpture. Understand "objet" or "objet de hideous" as the sculpture. The description of the sculpture is "It might possibly be natural, or it might be man-made. It might appeal to someone, but it is certainly not to your tastes."

Instead of showing the sculpture to Nathan:
    say "'Please put it anywhere,' he says."

Instead of giving the sculpture to Nathan:

say "'No, it doesn't work if I touch it. That's why I couldn't-- well, just put it down.'"

After dropping the sculpture:
now the player is aware of the breakage;
now the sculpture is nowhere;
say "You are incredibly careful, but somehow the sculpture slips -- you might almost say slithers -- from your fingers and crashes into a thousand shards at the feet of [Nathan].

There is a tense silence."

Before reading a command: now Nathan is active.

Every turn while not asking:
if Nathan is passive, rule succeeds;
if the player is carrying the sculpture:
if showing or giving, rule succeeds;
say "[Nathan] opens one eye and stares at you meaningfully. He is waiting for you to deposit his objet de hideous somewhere.";
rule succeeds;
if the breakage is pending:
if dropping, rule succeeds;
say "You're not quite sure where to begin, but you can't very well leave without making at least some remark on the smashing of the sculpture.";
rule succeeds;
if a subject is pending:
choose a random row in the Table of Offhand Reminiscences;
say "[line entry][paragraph break]".

Table of Offhand Reminiscences
line
"'It actually is kind of a funny story about [the random pending subject],' [Nathan] remarks casually."
"[Nathan] chuckles under his breath. 'Man, I hadn't

thought about [the random pending subject] in ages.'"
"He glances sideways at you. 'It's nothing personal, you know, but I don't feel comfortable discussing [the random pending subject] with just anyone.'"
"'I don't know why I brought up [the random pending subject] just now,' [Nathan] comments. 'Don't mention it to anyone, if you don't mind.'"
"'Okay, see, the thing about [the random pending subject] is...' [paragraph break]'Yes?' you ask, on cue. [paragraph break]'...never mind.'"
"[Nathan] makes an explosive exasperated sound. 'Don't you want to ask me about [the random pending subject]?' he demands."

Test me with "highlighting bright / put sculpture on pedestal / put it on table / put it on sickbed / drop it / think / ask nathan about breakage / think / ask him about generator / ask him about dreams / ask him about gift / ask him about shari / ask him about scar".

---

342 **ExampleHudsucker Industries** *RB*

Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

In this scenario, the player starts with a bag full of unsorted letters. These can be polite or rude, but he won't know which until he has examined them. What's more, he is allowed to sort the letters, in which case a group of letters will be shown as (for instance) "two polite letters"; but a group of mixed letters, even if they have all been read, will be called "unsorted letters".

Further, the player should be allowed to refer to sorted letters by tone, but not unsorted letters.

To do this, we'll need printing the name... and printing the plural name..., as well as some special understanding rules.

"Hudsucker Industries"

Tone is a kind of value. The tones are effusive, affectionate, polite, curt, and flamingly rude.

A letter is a kind of thing. The description of a letter is usually "On inspection, it turns out to be quite [tone]." A letter has a tone. The tone of a letter is usually polite.

A letter can be read or unread. A letter is usually unread. Carry out examining a letter: now the noun is read.

Before printing the name of a read letter: say "[tone] ".

Before printing the name of an ungrouped letter: say "random ".

Before printing the plural name of a letter (called the subject):
    if the subject is grouped:
        say "[tone] ";
    otherwise if the number of unread letters which are next to the subject is 0:
        say "unsorted ".

After printing the plural name of a letter (called the subject):
    if the number of read letters which are next to the subject is 0, say " (all unread, at the moment)" instead;
    if the number of unread letters which are next to the subject is greater than 0, say " (some as yet unread)" instead.

Proximity relates a thing (called X) to a thing (called Y) when the holder of X is the holder of Y. The verb to be

```
next to means the proximity relation.


Definition: a letter is grouped:
    if it is unread, no;
    if the number of unread letters next to it is greater
than 0, no;
    repeat with item running through letters which are
next to it:
        if the tone of item is not the tone of it, no;
    yes.


Definition: a letter is ungrouped if it is not grouped.


The Mailroom is a room. "Usually a thrumming hive of
bee-like workers, but you got in early to get a jump on
the day's work."


The satchel is carried by the player. Two flamingly rude
letters are in the satchel. Five polite letters are in the
satchel.


The mail wall are fixed in place in the mailroom. "Before
you is a wall of mailboxes, including [a list of mailboxes
which are part of the mail wall]."


The plural of mailbox is mailboxes. A mailbox is a kind
of container. The CEO box is a mailbox. The Hold box is
a mailbox. The Trash box is a mailbox. Understand
"mailbox" as a mailbox.
```

Now, there's a good bit of interaction to streamline. We intend that the player will be taking letters from the satchel, reading them, and putting them (perhaps grouped) into boxes. Our interaction rules should assist in this process as much as possible. To start with, the player will be most likely to examine letters he hasn't read yet:

```
Does the player mean examining a letter (called the
subject):
    if we have examined the subject, it is very unlikely;
    it is very likely.
```

The rules about taking are more subtle: the player is more likely to want to take an ungrouped letter than a grouped one; he is more likely to want one from the satchel than not; and he is most unlikely to want to take a letter (grouped or ungrouped) that he is already holding.

```
Does the player mean taking a letter (called subject)
which is grouped:
    if the player carries the subject, it is very unlikely;
    if the subject is in the satchel, it is possible;
    it is unlikely.
```

```
Does the player mean taking a letter (called subject)
which is ungrouped:
    if the player carries the subject, it is very unlikely;
    if the subject is in the satchel, it is very likely;
    it is possible.
```

And finally, we will assume by default that anything other than examining or taking is most likely to apply to a letter he's already identified:

```
Does the player mean doing something other than
examining or taking with a letter (called the subject):
    if we have examined the subject, it is likely;
    it is unlikely.
```

And we would also like to understand properties under the same circumstances as printing -- a letter will be identifiable as "polite" if it's already been read and it is either by itself or in a sorted stack of polite letters, but otherwise not. What's more, to make it possible to disambiguate commands in the other direction, we'll call any unsorted letter "random", to represent that the player doesn't know what it is.

```
Understand the tone property as referring to a letter
when the item described is grouped. Understand
```

```
"random" as a letter when the item described is
ungrouped.


When play begins:
    now every mailbox is part of the mail wall;
    repeat with switch count running from 1 to 5:
        move a random letter to the satchel.


Test me with "inventory / examine letter / get letter / i /
put letter in ceo box / inventory / get letter / x letter / g / g
/ i / x letter / g / g / i / put letter in hold box / get letter / g /
g / i".
```

That last "repeat" is merely a device to shuffle the order of items in the satchel so that the player will not always encounter the letters in a neatly presorted order, despite our defining them that way. (Of course, that means that the test produced by TEST ME cannot be very exciting...)

---

343 ExampleProlegomena *RB*

Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

Room descriptions often make the player character out to be a bit of a savant, able to count whole stacks of items at a glance: "You see 27 paper clips here."

We can adjust this behavior to our liking, though, with the printing a number... activity, as follows:

```
"Prolegomena"


The Editor's Office is a room. The desk is a supporter in
the Editor's Office.
```

```
A red pencil is a kind of thing. 12 red pencils are on the
desk.

A letter is a kind of thing. 12 letters are on the desk.
Understand "correspondence" as a letter.

Rule for printing the plural name of a letter:
    if the listing group size is greater than 7, say
"correspondence";
    otherwise say "letters".

Rule for printing a number of something (called the
target) when the listing group size is greater than 7:
    say "[one of]some [or]various [or]an assortment of [at
random]";
    carry out the printing the plural name activity with the
target.
```

This general rule can of course be overridden by more specific ones; for instance, if we want to take the opportunity to comment on the viewpoint character's appetite for instruments of correction:

```
Rule for printing a number of red pencils (called the
target) when the listing group size is greater than 10:
    carry out the printing the plural name activity with the
target;
    say " in nearly-sufficient quantity".

Test me with "get two letters / look / get a pencil / i / get
pencil / g / g / look / i / get all / i".
```

ExampleUnpeeled *RB*

Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

```
"Unpeeled"

Scullery is a room. A sack is carried by the player. The
sack contains a yellow onion. The player carries a cork.

Before printing the name of the onion while listing
contents:
    if the holder of the onion contains exactly 1 thing, say
"single ".

Test me with "i / put cork in sack / i".
```

---

345 ExampleWesponses *RB*

Parser messages that are delivered with a speech impediment.

If we want to change individual responses to player action, then the best thing to do is to use the response facility to modify those selections, as shown in the chapter on Responses.

However, suppose what we want is to give the parser a speech impediment that slightly alters all of the responses it issues. For this purpose, we might need to do a bit of text replacement...

```
"Wesponses"

The Office is a room. Barry Kripke is a man in the
Office.
```

```
The response inhibition is initially false.


Rule for issuing the response text of a response (called
R) when response inhibition is false:
    now response inhibition is true;
    let output be "[text of R]";
    now response inhibition is false;
    replace the text "r" in output with "w";
    replace the text "R" in output with "W";
    say "[output]".


Test me with "i / x barry / listen / waffle / jump"
```

Notice that this doesn't affect the printed names of objects in the room description or other kinds of output text -- only those that are issued by the response mechanism.

It would also not work to try to give our parser a nervous personality by simply adding "Um, " to the beginning of each response, because responses are not guaranteed to be full standalone sentences. For example, we could imagine writing

```
Rule for issuing the response text of a response (called
R) when response inhibition is false:
    now response inhibition is true;
    let output be "Um, [text of R]";
    now response inhibition is false;
    say "[output]".
```

but here is what the room description would say as a result:

```
Office
Um, You Um, can see Barry KripkeUm, here
```

ExampleRules of Attraction *RB*

A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

Often we have some salient features of an object that we want to make sure the player notices whenever looking at the item in a room or in inventory. At other times, we may prefer to allow the name of the item to be printed bare. So for instance:

```
"Rules of Attraction"

A metal form is a kind of thing. A magnet is a kind of
metal form.

Every turn:
    repeat with item running through nonmagnetic metal
forms which are not part of something:
        if item is in a container which contains a magnet
(called attractor):
            say "[The item] sticks to [the attractor].";
            now the item is part of the attractor.

The horseshoe magnet is a magnet carried by the
player. The nail is a metal form carried by the player.
The Barn is a room. In the Barn is a bucket. In the
bucket is a metal form called the iron hook.

Definition: a thing is nonmagnetic if it is not a magnet.

Rule for printing room description details of a magnet
(called attractor): if something is part of the attractor,
say " (stuck to which [is-are the list of things which are
part of the attractor])".

After printing the name of a magnet (called attractor)
while taking inventory:
    if something is part of the attractor, say " (stuck to
```

which [is-are the list of things which are part of the attractor])".

Before taking a touchable thing which is part of a magnet (called attractor):
    move the noun to the holder of the attractor.

Test me with "i / put horseshoe in bucket / look / get horseshoe / i / drop horseshoe / i / look / get all / put all in bucket / i / x magnet / get nail / i".

Example **Zorn of Zorna** *RB*

347

Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

"Zorn of Zorna"

Visibility rule:
    if examining:
        if the detail of the noun is fine and the number of visible lit candles is less than 5, there is insufficient light;
        if the detail of the noun is ordinary and the number of visible lit candles is less than 3, there is insufficient light;
    there is sufficient light.

Detail is a kind of value. The details are fine, ordinary, and gross. A thing has detail.

A candle is a kind of thing. Before printing the name of a candle while not burning or blowing out: say "[if lit]lit

[otherwise]unlit [end if]". A candle is usually lit. Before printing the plural name of a candle while not burning or blowing out: say "[if lit]lit [otherwise]unlit [end if]". A candle is usually lit. Understand the lit property as describing a candle. A candle is usually gross.

Instead of burning a candle: now the noun is lit; say "You light [the noun]."

Understand "blow out [something]" or "extinguish [something]" or "put out [something]" as blowing out. Understand the command "snuff" as "extinguish". Blowing out is an action applying to one thing.

Understand "burn [unlit candle]" as burning.

Instead of blowing out a candle:
    now the noun is unlit;
    say "You put out [the noun]."

Rule for printing a refusal to act in the dark:
    if we are examining something, say "The details of [the noun] are too fine to make out in the light of only [the number of visible lit candles in words] candle[s]." instead.

Every turn when the Todal is visible:
    if the number of visible lit candles is greater than 1:
        say "The brightness of the room wakens the Todal from slumber, and with you unarmed...";
        end the story;
    otherwise:
        say "Todal sleeps fitfully, troubled by even that faint light."

A room is usually dark.

The Palace is a room. "The Duke is out; the way is clear. East is Saralinda's Chamber; north, a hallway zigs and zags down to the gate that leads out." A finely-written placard is in the Palace. "A finely-written placard is on the wall next to this exit." The placard is fine. The description of the placard is "You read: 'Beware the Todal: its bite is worse than its gleep.

No more than one candle!'"

The candle-stand is a supporter in the Palace. Understand "stand" as the candle-stand. The description of the candle-stand is "The candle-stand is a tall metal branch for holding lights, but someone has quite practically added casters to the bottom." It is pushable between rooms. Three candles are on the candle-stand. Instead of removing something from the candle-stand: say "[The noun] is fixed quite firmly in place." Instead of taking something which is on the candle-stand: say "[The noun] won't come out of the holder." Instead of putting something on the candle-stand: say "[The candle-stand] is full."

Saralinda's Chamber is east of the Palace. "Now that Saralinda herself is gone, there is no real radiance in this place." Two unlit candles are in Saralinda's Chamber.

A large-print romantic novel is in Saralinda's Chamber. The novel is ordinary. The description of the novel is "'She Was Only The Chimney-Sweep's Daughter', by Marie Swelldon."

The Zig-Zag Hallway is north of the Palace. "The

Hallway goes left, then right, then left again..." Two unlit candles are in the Hallway.

Todal is an animal in the Zig-Zag Hallway.

Rule for printing the description of a dark room when the Todal is in the location:
    try listening.

Instead of listening when in darkness and the Todal is in the location:
    say "In the darkness something softly gleeps."

Instead of going north from the Hallway when in darkness:
    say "You stumble and cannot find your way."

North of the Hallway is Freedom. Instead of going to Freedom: say "You make it out into the cool night air at last!"; end the story finally.

Test me with "examine placard / get placard / n / listen / n / s / examine candle-stand / push candle-stand east / examine novel / get unlit candle / light it / light unlit candle / examine placard / push candle-stand west / e / examine novel / w / n / n".

ExampleHohmann Transfer *RB*

Changing the way dark rooms are described to avoid the standard Inform phrasing.

Inform automatically keeps track of light and darkness, handling such

questions as whether a room is lit, whether the player can see any light sources, etc., and then managing the descriptions accordingly. When the room is dark and no light sources are visible, the player is said to be "in darkness".

If we don't specify otherwise, Inform will describe our surroundings in a dark room thus:

```
Darkness
It is pitch dark, and you can't see a thing.
```

This is fine in many situations, but we may sometimes want to replace this phrase with something else.

```
"Hohmann Transfer"

The Western Hemisphere is a dark room. "The cloud
mass covers much of the land on this side of the planet,
and a particularly nasty storm is brewing off to the
south."

The Eastern Hemisphere is west of the Western
Hemisphere. The Eastern Hemisphere is east of the
Western Hemisphere. The Eastern Hemisphere is north
of the Western Hemisphere. The Eastern Hemisphere is
south of the Western Hemisphere. "This side of the
planet is more ocean than land, with only two continents
worthy of the name, and a volcanic archipelago in the
north seas."

Use full-length room descriptions.

Rule for printing the description of a dark room:
    say "It's night on this side of the planet, so you can
make out only the glow of urbanized areas along the
seacoasts." instead.
```

```
Rule for printing the name of a dark room:
    say "Dark Side" instead.
```

And now a few minor refinements so that we can see what happens when one room becomes dark and the other light:

```
Carry out going:
    say "You fire the thrusters and loop around to the
other side of the planet before settling into a new
geosynchronous orbit. Six months and one minute
later..."


The time of day is 4:55 PM.


At 5 PM:
    now the Eastern Hemisphere is dark;
    now the Western Hemisphere is lit.


Rule for printing the announcement of darkness:
    say "The planet abruptly spins itself over, exposing its
cool underbelly to the sun."


Test me with "e / z / z / w / z / z / e".
```

ExampleFour Stars 1 *RB*

An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead.

```
"Four Stars"
```

A thing has some text called sound. The sound of a thing is usually "silence".

The report listening rule is not listed in the report listening to rules.

Carry out listening to something:
    say "From [the noun] you hear [the sound of the noun]."

Instead of listening to a room:
    if an audible thing can be touched by the player, say "You hear [the list of audible things which can be touched by the player].";
    otherwise say "A merciful peace prevails."

Definition: a thing is audible if the sound of it is not "silence".

Before printing the name of something audible while listening to a room:
    say "[sound] from the "

A thing has some text called scent. The scent of a thing is usually "nothing".

The report smelling rule is not listed in the report smelling rulebook.

Carry out smelling something:
    say "From [the noun] you smell [scent of the noun]."

Instead of smelling a room:
    if a scented thing can be touched by the player, say "You smell [the list of scented things which can be touched by the player].";
    otherwise say "The place is blissfully odorless."

Definition: a thing is scented if the scent of it is not "nothing".

Before printing the name of something scented while smelling a room: say "[scent] from the "

The Waning Moon Resort is a dark room. "A spacious room with a flagstone floor, and a dreamcatcher hung over the king-size bed." The dreamcatcher is scenery in the Resort. The description is "The usual web of threads and crystals, feathers and beads." Instead of taking the dreamcatcher, say "Ah, ah -- you might be tempted to take it as a souvenir, except that the price list in the minibar clearly states they charge $65 apiece if you walk off with one. Cheaper than stealing the Frette bathrobes, but still probably not a good idea."

The king-size bed is an enterable supporter in the Resort. The description is "200-thread-count Egyptian cotton sheets, according to the website. You would make fun, only they really are extraordinarily comfortable." The player is on the bed. A Lindt chocolate is on the bed. It is edible. The scent of the chocolate is "chocolate-hazelnut smell".

Instead of exiting: say "You are too weary to move."

The suitcase is an openable closed container in the Resort.

An electric light is a kind of device. Carry out switching on an electric light: now the noun is lit. Carry out switching off an electric light: now the noun is unlit. Understand "light" as an electric light.

The solar lamp is an electric light in Waning Moon Resort. The description is "Specially designed to give light in a spectrum resembling sunlight, to improve the mood and make a person energetic." The lamp is switched on and lit.

An electric noisemaker is a kind of device. An electric noisemaker has some text called usual sound. The usual sound of an electric noisemaker is usually "beepbeepbeep". Carry out switching on an electric noisemaker: now the sound of the noun is the usual sound of the noun. Report switching on an electric noisemaker: say "[The noun] goes [usual sound of the noun]!" instead. Report switching off an electric noisemaker: say "You switch off [the noun], silencing the [usual sound of the noun]." instead.

Carry out switching off an electric noisemaker: now the sound of the noun is "silence".

The bedside table is in the Resort. The table supports a potted plant and a Bose speaker. The scent of the potted plant is "rosemary"

The Bose speaker is an electric noisemaker. The usual sound of the speaker is "soothing whalesong". The

sound of the speaker is "soothing whalesong". The speaker is switched on.

Instead of touching a device: say "You feel the surface of [the noun] and discover the switch."

Instead of touching a scented thing: say "The brush of your fingers stirs loose a fresh cloud of [scent of the noun] smell."

Rule for printing the description of a dark room: try listening; try smelling; rule succeeds.

Instead of examining an audible thing while in darkness: try listening to the noun. Instead of examining something while in darkness: try touching the noun.

Before touching something when in darkness:
    say "You grope about..."

After inserting the plant into something:
    say "You unceremoniously dump [the noun] into [the second noun], hoping it sustains no important damage thereby."

Before printing the name of a dark room: if the player can touch an audible thing, say "Noisy "; if the player can touch a scented thing, say "Perfumed ".

Visibility rule when in darkness:
    if examining something, there is sufficient light;
    there is insufficient light.

Rule for printing the announcement of darkness: say "It is now pleasantly lightless in here." instead.

Rule for deciding the scope of the player while in darkness: place the location in scope.

To decide whether in daylight:
    if in darkness, no;
    yes.

Instead of sleeping when in daylight:
    say "You've never been able to sleep with the light on."

Instead of sleeping when the player can touch an audible thing (called the irritant):
    say "The steady [sound of the irritant] from [the irritant] prevents your slumber."

Instead of sleeping when the player can touch a scented thing (called the irritant):
    if the irritant is chocolate, say "The smell of chocolate continues to tantalize you, keeping you from sleep.";
    otherwise say "You sniffle. [The irritant] is probably acting on your allergies."

Instead of sleeping:
    say "You slip easily into the arms of Morpheus.";
    end the story finally saying "At last..."

When play begins:
    say "You have at last escaped from the airport and

gotten through customs; survived an unnerving taxi ride over icy highways; stared down the impertinent concierge; endured the bellhop's catalog of features in your room; and achieved, finally, a moment of peace. Time for a good night's slumber!"

Test me with "x dreamcatcher / switch lamp off / look / sleep / eat chocolate / sleep / get plant / examine plant / open suitcase / put plant in suitcase / close suitcase / sleep / look / examine bose / switch bose off / sleep".

---

350 **Example** **Ways Out** *RB*

A status line that lists the available exits from the current location.

A not-uncommon device in games with large maps is a list of available exits printed in the status bar. We might do this so:

```
"Ways Out"

When play begins:
    now left hand status line is "Exits: [exit list]";
    now right hand status line is "[location]".

To say exit list:
    let place be location;
    repeat with way running through directions:
        let place be the room way from the location;
        if place is a room, say " [way]".
```

We may find that printing out full directions makes the status line unpleasantly crowded. Fortunately, it isn't hard to provide a set of abbreviations to use in this context:

```
Rule for printing the name of a direction (called the way)
while constructing the status line:
    choose row with a heading of the way in the Table of
Abbreviation;
    say "[shortcut entry]".


Table of Abbreviation

heading                 abbreviation
"north"                 "N"
"northeast"             "NE"
"northwest"             "NW"
"east"                  "E"
"southeast"             "SE"
"south"                 "S"
"southwest"             "SW"
"west"                  "W"
"up"                    "U"
"down"                  "D"
"inside"                "IN"
"outside"               "OUT"


Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "w / n / e / e / s / e".
```

Everywhere else, the names of directions will still be printed out in full in the usual way.

A status line that lists the available exits from the current location, changing the names of these exits depending on whether the room has been visited or not.

It may sometimes be helpful to prompt the player with a list of exits printed up in the status line. For instance, here is a status line that will print the names of nearby rooms, as well as all the doors the player can see:

```
"Guided Tour"

When play begins:
    now left hand status line is "Nearby: [if a room is
adjacent][the list of adjacent rooms][end if][if a room is
adjacent and a door is visible] and [end if][if a door is
visible][the list of visible doors][end if]";
    now right hand status line is "".
```

Of course, we may not want to tell the player what glories are to be found in locations he hasn't yet explored.

```
Rule for printing the name of an unvisited room (called
the target) while constructing the status line:
    let aim be the best route from the location to the
target;
    say "something [aim]".
```

Even when we have seen a room, we might still want a reminder about how to get there:

```
After printing the name of a visited room (called the
target) while constructing the status line:
    let aim be the best route from the location to the
target;
    say " ([aim])".
```

We may also find that printing out full directions makes the status line

unpleasantly crowded. Fortunately, it isn't hard to provide a set of abbreviations to use in this context:

```
Rule for printing the name of a direction (called the aim)
while constructing the status line:
    choose row with a heading of the aim in the Table of
Abbreviation;
    say "[shortcut entry]".


Table of Abbreviation

heading                 shortcut entry
"north"                 "N"
"northeast"             "NE"
"northwest"             "NW"
"east"                  "E"
"southeast"             "SE"
"south"                 "S"
"southwest"             "SW"
"west"                  "W"
"up"                    "U"
"down"                  "D"
"inside"                "in"
"outside"               "out"
```

Everywhere else, the names of directions will still be printed out in full in the usual way. And now we give it a little map to work with:

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The church door is east of Eastern End and west of the
```

```
Courtyard. The church door is a door.


Test me with "n / w / s".
```

Note that while this looks fine in some places, other locations exceed the limits of what the status-line can hold: if any given room is going to have a large number of exits, this kind of listing will almost certainly not fit. So apply cautiously.

---

352 **Example** **Reflections** *RB*

Emphasizing the reflective quality of shiny objects whenever they are described in the presence of the torch.

```
"Reflections"


Behind the Waterfall is a room. "Though one wall of the
cave is open to the waterfall, the quantity of water is so
great that barely any light comes through from the
outside." Behind the Waterfall is dark.


Surface is a kind of value. The surfaces are shiny and
dull. A thing has a surface. A thing is usually dull.


The player carries a reflecting ball, a canopic jar, an
abacus, a plumbline, a piece of chalk, and a torch. The
reflecting ball is shiny.


Aladdin's lamp is a shiny thing in Behind the Waterfall.
```

Brightness is a kind of value. The brightnesses are guttering, weak, radiant and blazing. The torch has a brightness. The torch is blazing. The torch is lit.

Understand "blow out [something]" or "blow [something]" or "extinguish [something]" as blowing out. Blowing out is an action applying to one thing.

Carry out blowing out: say "Futile."

Instead of blowing out the torch:
    now brightness of torch is the brightness before the brightness of the torch;
    say "The light of the torch dies to [brightness of torch]."

Instead of blowing out the guttering torch:
    say "Fool! Do you want to put it out entirely?"

Rule for writing a paragraph about a shiny thing:
    say "The [brightness of the torch] light of [the torch] reflects in the surface[if the number of shiny things in the location > 1]s[end if] of [the list of shiny things in the location]."

Before printing the name of the torch while writing a paragraph about something:
    if the torch is in the location, say "fallen ".

Test me with "drop ball / look / blow torch / look / drop torch / look".

Social dynamics in which groups of people form and circulate during a party.

To start with, let's understand "room" to mean "a group of people talking". These groups can grow and shrink as people come and go, so we'll want to name and rename them; and we're also going to need some rules to motivate people moving around, and a description to narrate how they behave when we're with them.

```
"Emma"

by the banquet table is a room. at the corner is a room.
next to the doorway is a room. by the window is a room.

Social clump is a kind of value. The social clumps are
vacancy, lone person, couple, cluster, group.

A room has a social clump. Understand the social clump
property as describing a room.

Before printing the name of a room:
    say "a [social clump] ".

After looking:
    assign clumping;
    say "Elsewhere in the room, you can see [the list of
rooms which are not the location]."

Understand "go to [any room]" as joining. Joining is an
action applying to one visible thing. Carry out joining:
move player to the noun. Report joining: do nothing.
```

```
Understand "examine [any room]" as looking toward.
Looking toward is an action applying to one visible thing.
Carry out looking toward a room: say "In that direction
you see [a list of other people in the noun]."


When play begins: assign clumping. Every turn: assign
clumping.


To assign clumping:
    repeat with space running through rooms:
        now the social clump of the space is vacancy;
        if the space contains exactly 1 person, now the
social clump of the space is Lone person;
        if the space contains exactly 2 people, now the
social clump of the space is Couple;
        if the space contains more than 2 people and the
space contains fewer than 5 people, now the social
clump of the space is cluster;
        if the space contains more than 4 people, now the
social clump of the space is group.


The room description heading rule is not listed in the
carry out looking rules.


A person has a number called longevity. The longevity
of a person is usually 0. A person can be active or
passive.


Definition: a person is other if it is not the player.


Every turn:
    repeat with mover running through other people:
        now the mover is active;
        increment the longevity of mover;
```

```
        if longevity of mover is greater than 3 or the mover
is bored:
            assign value of spaces for the mover;
            let destination be the nicest room;
            if the destination is not the location of the mover:
                if the player can see the mover, say "[The
mover] makes excuses and drifts off to join [the
destination].[paragraph break]";
                move the mover to the destination;
                now the mover is complacent;
                now the longevity of the mover is 0;
                if the player can see the mover, say "[The
mover] wanders over.[paragraph break]";
                assign clumping;
                now mover is passive.



A room has a number called attractiveness.



Definition: a room is nice if its attractiveness is 1 or
more.



To assign value of spaces for (mover - a person):
    repeat with space running through rooms:
        now attractiveness of the space is 0;
        repeat with figure running through people in the
space:
            if the mover is bored, decrease attractiveness of
the space by 2;
            if the mover likes the figure, increment
attractiveness of the space;
            if the mover dislikes the figure, decrement
attractiveness of the space;
            if the mover desires the figure, increase
attractiveness of the space by 2.



Liking relates various people to various people. The
verb to like means the liking relation.



Disliking relates various people to various people. The
```

verb to dislike means the disliking relation.

Attraction relates various people to various people. The verb to desire means the attraction relation.

Mr Weston, Mr Woodhouse, Mr Elton, Mr Knightley, and Frank Churchill are men. Mrs Weston, Mrs Bates, Miss Bates, Harriet Smith, Emma Woodhouse, and Jane Fairfax are women.

Harriet Smith likes Mr Elton. Harriet Smith desires Mr Elton. Harriet Smith likes Emma Woodhouse.

Mr Elton desires Emma Woodhouse.

Emma Woodhouse likes Harriet Smith and Mr Knightley. Emma Woodhouse dislikes Jane Fairfax.

Mr Knightley likes Emma Woodhouse, Mr Weston, and Mrs Weston. Mr Knightley desires Emma Woodhouse.

Jane Fairfax desires Frank Churchill. Jane Fairfax likes Frank Churchill.

Frank Churchill desires Jane Fairfax and Emma Woodhouse. Frank Churchill likes Jane Fairfax.

Miss Bates likes Jane Fairfax, Emma Woodhouse, and Mrs Bates.

```
Mr Weston likes Frank Churchill, Emma, Knightley, and
Mrs Weston.


Mrs Weston likes Frank Churchill, Emma, Knightley, and
Mr Weston.


Mrs Bates likes Miss Bates.


A person can be complacent or bored.


When play begins:
    repeat with character running through other people:
        let space be a random room;
        move character to space.
```

And now we use writing a paragraph about... to describe character behavior in groups, when we join them:

```
Rule for writing a paragraph about Frank Churchill:
    if the location contains a woman (called flirt) who is
desired by Frank:
        say "[Frank Churchill] is talking with great
animation and slightly more than becoming warmth to
[the flirt][if an unmentioned other person is in the
location], while [the list of unmentioned other people in
the location] look on with varying degrees of amusement
or irritation[end if].";
        repeat with character running through people in the
location:
            if the character is not Churchill and the character
is not the flirt, now the character is bored.


Rule for writing a paragraph about Mr Elton:
    if the location contains an unmentioned woman
```

```
(called flirt) who is desired by Elton:
        say "[Mr Elton] hangs on the sleeve of [the flirt],
offering an assortment of studied gallantries that make
you wonder about his good sense.";
        repeat with character running through people in the
location:
            if the character is not Elton and the character is
not the flirt, now the character is bored.


Rule for writing a paragraph about Harriet Smith:
    if the location contains Emma and Emma is
unmentioned:
        say "[Harriet] and [Emma] are conversing in low
tones -- Harriet, apparently, being too shy to speak so
that everyone can hear her."


Rule for writing a paragraph about Mr Knightley:
    if the location contains an unmentioned man (called
the listener) who is not Mr Knightley:
        say "[Mr Knightley] is speaking with [the listener]
about agricultural matters.";
        now the listener is complacent.


Rule for writing a paragraph about Miss Bates:
    say "[Miss Bates] is giggling about the weather[if an
unmentioned other person is in the location]. This does
not seem to compel the interest of [the list of
unmentioned other people in the location][end if].";
    repeat with character running through people in the
location:
        if the character is not Miss Bates and character is
not Mrs Bates, now the character is bored.
```

Since this is just an example, we'll stop here, but there's no reason we couldn't write such paragraphs for everyone.

```
Test me with "z / z / z / look / x corner / x doorway / x
window / x table / go to the table".
```

ExampleAir Conditioning is Standard *RB*

Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

```
"Air Conditioning is Standard"


Section 1 - The Garage


A person has some text called current occupation. The
current occupation of a person is usually "None".


Mood is a kind of value. The moods are bemused,
bored, attentive, rapt, and blushing. A person has a
mood. A person is usually attentive.


Instead of examining a person:
    now every thing is unmentioned;
    carry out the writing a paragraph about activity with
the noun.


Rule for writing a paragraph about a person (called X):
    let the subsequent mention be "Name";
    if the current occupation of X is not "None":
        say "[current occupation of X]. ";
        let the subsequent mention be "He";
        if X is female, let the subsequent mention be "She";
    if X wears something unmentioned:
        if the subsequent mention is "Name", say "[The X]
";
        otherwise say "[subsequent mention] ";
        say "is wearing [a list of unmentioned things worn
by X]";
        if X carries something unmentioned, say " and
```

carrying [a list of unmentioned things carried by X]";
        say ".";
    otherwise:
        if X carries something unmentioned:
            if the subsequent mention is "Name", say "[The X] ";
            otherwise say "[subsequent mention] ";
            say " is carrying [a list of unmentioned things carried by X]."

The Garage is a room. "Above the street door is a spectacular art nouveau fanlight, wherein a stained-glass Spirit of Progress bestows the gift of Transportation on mankind.

The sun, gleaming through the hair of Progress, throws amber curls on the macadam floor."

The fanlight is scenery in the Garage. The description is "A semi-circle of stained glass as wide as the garage door, designed by Louis Comfort Tiffany himself. No expense has been spared."

The gift of Transportation is part of the fanlight. The description is "The gift of Transportation is envisioned as a cornucopia disgorging a steam locomotive. And that blue bit of glass might be the Montgolfier balloon."

The Spirit of Progress is part of the fanlight. The description is "It is part of her character to have bare shoulders like that."

The machinist is a bored man. He is in the Garage. He is wearing a grimy pair of overalls. He carries a wrench and a screwdriver. The current occupation of the machinist is "[The machinist] is making some

adjustments to [the random thing which is part of the Victorian Car] with his [random thing carried by the machinist]"

The Victorian Car is a device in the Garage. A cast-iron steering wheel, a leather bucket seat, a horn, and a combustion engine are part of the Victorian Car. The seat is an enterable supporter.

```
Rule for writing a paragraph about a device (called X):
    let the subsequent mention be "Name";
    if the X is unmentioned:
        say "[The X] is here. ";
        let the subsequent mention be "It";
    if something is part of X:
        if the subsequent mention is "Name", say "[The X]
";
        otherwise say "[subsequent mention] ";
        say "[if a mentioned thing is part of X]also [end
if]features[if a mentioned thing is part of X], in addition to
[the list of mentioned things which are part of X],[end if]
[a list of unmentioned things which are part of X]";
        say ".".
```

```
Rule for printing the name of the steering wheel while
writing a paragraph about a person:
    say "steering wheel".
```

A supporter has some text called position. The position of a supporter is usually "None".

The Office is west of the Garage. The Office contains a desk. The desk has the position "A [desk] with several dozen drawers stands in the center of the room". On the desk are some papers.

```
After printing the name of a supporter (called X) which
supports an unmentioned thing:
    now X is unmentioned.



Rule for writing a paragraph about a supporter (called
X):
    let the subsequent mention be "Name";
    if the position of X is not "None":
        say "[position of X]. ";
        let the subsequent mention be "It";
    if a mentioned thing is on X:
        say "Besides [the list of mentioned things which
are on X], ";
        let the subsequent mention be "it";
    if the subsequent mention is "Name", say "[The X] ";
    otherwise say "[subsequent mention] ";
    say "holds [a list of unmentioned things which are on
X]."



Section 2 - Schedule



The time of day is 4:38 PM.



At 4:42 PM:
    move the machinist to the Office;
    say "The machinist wanders into the Office to get
some paperwork.";
    now every thing carried by the machinist is on the
desk;
    now the current occupation of the machinist is "[The
machinist] rifles through [the papers] on [the desk]".



At 4:43 PM:
    move the young lady to the Garage;
    if the young lady can be seen by the player,
        say "An attractive young lady walks in from the
street, and glances around as though she has never
been here before."
```

At 4:45 PM:
    if the young lady can be seen by the player,
        say "With a not-quite-convincing air of innocence, [the young lady] happens to lean upon [the horn], which bleats loudly.";
    otherwise say "There is a honk from the Garage[if the machinist can be seen by the player]. The machinist looks up with a frown[end if].";
    now the horn is mentioned.

At 4:46 PM:
    move the machinist to the Garage;
    say "The machinist strolls from the Office into the Garage to find out what is going on.";
    now the current occupation of the machinist is "[The machinist] is chatting with [the young lady]. He seems to be demonstrating the various features of [the car], including [the random thing which is part of the car]";
    now the current occupation of the young lady is "[The young lady] is asking [the machinist] a number of questions about [the car]".

At 4:49 PM:
    if the young lady can be seen by the player, say "[The machinist] gives [the young lady] his arm to climb into [the seat].";
    move the young lady to the seat;
    now the young lady is rapt;
    now the current occupation of the young lady is "[The young lady] is turning [the steering wheel] from side to side";
    now the current occupation of the machinist is "[The machinist] is leaning on the door of [the car], pointing out features to [the young lady]";
    move the besotted expression to the machinist;
    now the machinist is wearing the besotted expression.

At 4:52 PM:
    now the sober grey gown is unbuttoned at the neck;
    if the young lady can be seen by the player, say "[The

young lady] murmurs something about the wilting heat, and undoes a button or two of her gown. The machinist's expression is comical, or would be, if you weren't annoyed."

Every turn when the player is in the Garage and young lady is on the seat:
    say "You are beginning to feel a little unnecessary in this scene."

Every turn when the player is in the Office and the young lady is on the seat:
    say "There's no sound at all from the other room, not even conversation."

Before going to the Garage when the young lady is on the seat:
    now the sober grey gown is tellingly dishevelled;
    move the young lady to the Garage;
    now the young lady is blushing;
    say "There is a flurry of movement as you enter the room.";
    now the current occupation of the young lady is "[The young lady] stands near the door, tapping her foot nervously";
    now the besotted expression is nowhere;
    now the current occupation of the machinist is "[The machinist] is leaning against [the car], looking smug".

Section 3 - Initially Out of Play

The besotted expression is a wearable thing. The description is "It looks foolish, doesn't it?"

The young lady is a bemused woman. She is wearing a sober grey gown and a pair of black boots. The current

occupation of the young lady is "[The young lady] is running a gloved finger along the chassis of [the victorian car]"

Before printing the name of the young lady while writing a paragraph about a person:
    say "[mood of the young lady] "

The description of the grey gown is "Something about the perfect row of tiny buttons has the wrong effect -- at any rate, it is natural to wonder how long they take to undo." The gown can be buttoned almost to the chin, unbuttoned at the neck, or tellingly dishevelled.

Rule for printing the name of the gown when writing a paragraph about a person:
    say "sober grey gown ([sober grey gown condition])"

Test me with "z / look / look / z / look / west / east / z / look / z / look / z / look / west / east".

---

355 Example Rip Van Winkle *RB*

A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

There are times when, for greater elegance of prose, we'd like to mention an object in the main body text of a room. For instance:

Here is a lovely, secluded fold in the mountains, far from civilization: as though to prove it, Rip Van Winkle is sleeping under a tree.

As we've already seen, that's no problem if Rip is scenery. He'll stay there motionless.

But what if something in the game allows Rip to wake up? Or what if we want to use the same technique on a portable object that the player should be allowed to take? Clearly in that case it's not appropriate to make the mentioned thing be scenery, and at the same time, we need to keep Inform from adding a superfluous

```
You can see Rip Van Winkle here.
```

to the end of our description.

Here is how:

```
"Rip Van Winkle"


A person can be asleep.


The Catskills is a room. "Here is a lovely, secluded fold
in the mountains, far from civilization[if Rip Van Winkle
is asleep]: as though to prove it, Rip Van Winkle is
sleeping under a tree[end if]."


A tree is scenery in the Catskills.


Rip Van Winkle is a man in the Catskills. Rip Van
Winkle is asleep.


Before listing nondescript items of the Catskills:
    if Rip Van Winkle is marked for listing:
        now Rip Van Winkle is not marked for listing;
        if Rip Van Winkle is not asleep,
            say "Rip Van Winkle stands here, looking
mightily confused."
```

```
Instead of waiting:
    say "Rip Van Winkle wakes up with a snort.";
    now Rip Van Winkle is not asleep.


Test me with "look / z / look".
```

ExampleHappy Hour *RB*

Listing visible characters as a group, then giving some followup details in the same paragraph about specific ones.

Often it is best to have an entire paragraph about the characters present in a room, but suppose we're narrating a large party with a lot of people moving around. In that case, it might be better to list everyone together, then add a few salient details by way of follow-up, like this:

```
"Happy Hour"


Before listing nondescript items:
    say "You can see [a list of people who are marked for
listing] here. ";
    repeat with named party running through people:
        now the named party is not marked for listing;
    let count be the number of visible other people who
are carrying something;
    if count is 0:
        say paragraph break;
        continue the action;
    let index be count;
    repeat with holder running through visible other
people who are carrying something:
        if index is count, say "[The holder]";
        otherwise say "[the holder]";
        say " has [a list of things carried by the holder]";
        decrement index;
        make delimiter index of count.
```

The next part could be simpler, but for rigor we will write it in such a way that it will work whether or not the serial comma is in use. This requires

some extra work.

```
To make delimiter (index - a number) of (count - a
number), continuing or halting:
    if index is 0:
        if continuing, say ". [run paragraph on]";
        otherwise say ".";
    otherwise if index is 1:
        if count is 2, say " and ";
        otherwise say "[optional comma] and ";
    otherwise:
        say ", ".


To say optional comma:
    if the serial comma option is active:
        say ",".
```

And now the scene:

```
The Banquet Hall is a room. "A large cheery banner
over the door (which, incidentally, vanishes when you
approach it) reads: HELLO NEW INDUCTEES!
WELCOME TO THE AFTERLIFE!"


Fred, George, and Larry are men in the Banquet Hall.
Fred carries a dry martini. Larry carries a cream puff.
Matilda and Louise are women in the Banquet Hall.


Definition: a person is other if it is not the player.


Every turn:
    let wanderer be a random other person;
    let place be the holder of the wanderer;
    let next place be a random room adjacent to the
place;
    let the way be the best route from the place to the
next place;
```

```
try the wanderer going the way.
```

```
The Kitchen is west of the Banquet Hall. "Dominated by
a pile of dirty plates which you imagine it will be
someone's privilege to wash, later." Vanessa is a
woman in the Kitchen. Vanessa carries a tray. On the
tray is a salmon roll. The roll is edible.
```

```
Test me with "z / look / g / g / g ".
```

Example **The Eye of the Idol** *RB*

A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

```
"The Eye of the Idol"
```

```
Section 1 - Reusable Material
```

We start by defining relations that let us know where items "belong", with the understanding that if something is where it belongs, it will be described in the main room description and therefore should not be separately listed. Thus:

```
Positioning relates various things to various things. The
verb to be placed in means the positioning relation. The
verb to be placed on implies the positioning relation.
```

```
Room-positioning relates various things to various
rooms. The verb to be room-placed in means the room-
positioning relation.
```

We can't make relations relate various objects to various objects, and rooms are not things, so two separate cases are necessary. An alternative approach would be to say "A thing has an object called the initial placement", which would allow a thing to have an initial placement that was a room, a supporter, or a container; an advantage of using relations, though, is that that way we can if we like specify multiple placements for the same object, so that, e.g., a sparkling diamond can be described in the main description paragraph as "half-buried in dust" in the beginning of the game, and then at the end as "in the eye of the idol" at the end.

Now we define, based on these relations, an "in-place" adjective, which will identify whether something is in a location which will specially describe it:

```
Definition: a thing (called prop) is in-place:
    if the prop is in the location and the prop is room-
placed in the location, yes;
    if the holder of the prop is a thing and the prop is
placed in the holder of the prop, yes;
    no.


Definition: a thing is out-of-place if it is not in-place.
```

With that done, removing these items automatically from the room description is actually pretty easy:

```
Before listing nondescript items:
    now every marked for listing in-place thing is not
marked for listing.
```

One tricky case remains: when something is placed on a supporter that is scenery, it can be mentioned even if we have marked that object "not marked for listing". What matters here is not whether the object itself is marked for listing but whether the supporter has been "mentioned". (A fuller description of how room descriptions are assembled is available in the Looking section of the Commands chapter in the Recipe Book.) So let's also add a feature whereby we can easily suppress the descriptions of these supporters when appropriate:

```
A supporter can be quiet.
```

A quiet supporter is one that is never mentioned itself and which only mentions its contents if they are out of place. This allows for maximum flexibility in incorporating it into the body of room descriptions.

```
Rule for writing a paragraph about a quiet supporter
(called chosen table):
    if an out-of-place thing is on the chosen table:
        if an in-place thing is on the chosen table,
            say "On [the chosen table], in addition to [the list
of in-place things on the chosen table], [is-are a list of
out-of-place things which are on the chosen table].";
        otherwise say "On [a chosen table] [is-are a list of
out-of-place things which are on the chosen table].";
    now the chosen table is mentioned.
```

Notice that we can still override this with writing a paragraph rules about specific supporters in our game, if we decide that we want something a little different in some cases.

Now, an example to test this out:

```
Section 2 - A Sample Scenario
```

```
The Sand-Floored Chamber is a room. "The constant
wind has filled this chamber with a layer of fine red
sand, as soft as powder snow[if the diamond is in the
Sand-floored Chamber]. Something sparkling is half-
buried in the corner[end if]. A doorway lies open to the
north."
```

```
The sparkling diamond is in the Sand-floored Chamber.
The sparkling diamond is room-placed in the Sand-
floored Chamber. The description is "It is a vast
diamond; the front is faceted, the back smoothed to fit in
some sort of socket."
```

```
The Hexagonal Temple is north of the Sand-Floored
Chamber. "The temple walls are great ashlar blocks
rising to a hundred feet overhead, perhaps more; the
```

```
roof is a scarlet awning only, through which the sun
filters down in blood hues. Overseeing all is a sculpture
in stone and ivory[if the sparkling diamond is in the idol's
eye], in whose single eye a vast diamond gleams[end if]
[mat-and-incense text].".
```

```
To say mat-and-incense text:
    if the mat is in the Temple and the incense stick is on
the pedestal:
        say ". A prayer mat at the idol's feet, and an
incense stick still burning on the pedestal, indicate that
someone was only recently consigning her grievances
to the care of the deity";
    otherwise if the mat is in the Temple:
        say ". At the idol's feet, some worshipper has left a
prayer mat";
    otherwise if the incense stick is on the pedestal:
        say ". At the idol's side is a pedestal, on which
incense still smolders".
```

We could have done all this with text conditions in the main room description, but it becomes difficult to read when there are too many conditions operating in the same text property, so we break it out into a clearer set of conditions.

```
The idol is scenery in the Hexagonal Temple.
Understand "sculpture" or "stone" or "ivory" as the idol.
The description is "The idol is perhaps three times the
height of an ordinary man."
```

```
The idol's eye is part of the idol. It is a container. The
description is "[if the diamond is in the idol's eye]It
gleams with purpose and righteous wrath[otherwise]A
round socket in the center of the idol's forehead from
which something seems to be missing[end if]."
```

```
The pedestal is a quiet supporter in the Hexagonal
Temple. On the pedestal is an incense stick. The
incense stick is placed on the pedestal.
```

A mat is in the Hexagonal Temple. It is room-placed in the Hexagonal Temple. The description is "Woven of assorted grasses."

Test me with "get diamond / look / n / get mat / look / drop diamond / look / get diamond / put diamond in eye / look / get incense / look / drop mat / look / get mat / put mat on pedestal / look / put incense on pedestal / look".

---

358 **Example** **Priority Lab** *RB*

A debugging rule useful for checking the priorities of objects about to be listed.

When it comes time to start manipulating the priorities of items, it is useful to be able to check the table for debugging purposes; the problem is that printing the names of the objects can itself affect the way the room description is generated, foiling our debugging efforts.

What follows is a rule to help with debugging safely, and a sample of how priorities work:

```
"Priority Lab"

Section 1 - Procedure

Before printing the locale description (this is the dump
locale table rule):
    say "Locale Priority list:";
    repeat through Table of Locale Priorities:
        let the flag be whether or not the notable-object
entry is mentioned;
        say "[line break] [notable-object entry]: [locale
description priority entry]";
        if the flag is false, now the notable-object entry is
not mentioned;
    say line break.
```

Now, let's look at some items put in a specific order. Things with low priority numbers list towards the beginning; things with high priority numbers list towards the end. (It helps to think of it as though we were making a numbered list of the paragraphs to appear in the description.) Anything numbered 0 doesn't appear at all, and the default priority of an object is 1.

```
A thing can be early-described, late-described, latest-
described, never-described, sightline-described, or
ordinarily-described. A thing is usually ordinarily-
described.


After choosing notable locale objects (this is the apply
early and late description rule):
    repeat with item running through early-described
things:
        if there is a notable-object of item in the Table of
Locale Priorities:
            set the locale priority of the item to 1; [list before
everything else -- this would work with any number
lower than 5 and higher than 0]
    repeat with item running through late-described
things:
        if there is a notable-object of item in the Table of
Locale Priorities:
            set the locale priority of the item to 10; [list after
everything else -- this would work with any number
larger than 5]
    repeat with item running through never-described
things:
        set the locale priority of the item to 0; [don't list at
all]
    continue the activity.
```

An important cautionary note: priorities are only honored if the objects are going to get their own paragraphs (with "writing a paragraph about..." or because they have initial appearances). Priorities do not affect the order in which items appear in the final "You can see..." list, except that items with priority 0 or lower are omitted. (If we want to order the items in that list, we may want to resort to the Complex Listing extension by Emily Short.)

There are further refinements available to us: for instance, we could make some things that are only visible if the player is raised above ground level.

```
After choosing notable locale objects (this is the
sightline-described things are visible from supporters
rule):
    if the player is not on a supporter:
        repeat with item running through sightline-
described things:
            if there is a notable-object of item in the Table of
Locale Priorities:
                set the locale priority of the item to 0; [remove
objects that can only be seen from higher objects.]
    continue the activity.
```

It may also be useful to know about the "parameter-object", which refers to the thing whose contents we are currently describing: the standard rules consider how to describe the contents of the location and then also check the contents of any supporter or container the player may be inside, so in the first case "parameter-object" would be the location, and then in the second the supporter in question.

In practice this is rarely useful, but should we need to change priorities in the case of both player and object being inside a particular container, we might make use of it, for instance:

```
A thing can be tasteful or icky. A thing is usually tasteful.
```

```
After choosing notable locale objects (this is the icky
things next to players rule):
    if the player is on the parameter-object:
        repeat with item running through icky things :
            if there is a notable-object of item in the Table of
Locale Priorities:
                set the locale priority of the item to 10;
[remove objects that can only be seen from higher
objects.]
    continue the activity.
```

The other thing to note is that by default that final collection of generic objects ("You can also see...") appears at the end, regardless of the priority of everything else. If we really wanted to, though, we could force something to appear even after that paragraph, by adding a new listing rule to the locale description rules:

After choosing notable locale objects (this is the latest-described items priority rule):
    repeat with item running through latest-described things:
        if the item is a notable-object listed in the Table of Locale Priorities:
            now the item is mentioned;
            now the item is marked for late listing.

The late listing rule is listed after the you-can-also-see rule in the for printing the locale description rules.

A thing can be marked for late listing. A thing is usually not marked for late listing.

This is the late listing rule:
    if something is marked for late listing:
        say "Oh! And also [a list of things which are marked for late listing].";
        now everything is not marked for late listing;
    continue the activity.

Section 2 - Scenario

The Priority Lab is a room. The early bird, the worm, the leaf, the unseen object, the pebble, the twig, and the late edition are things in the Priority Lab.

The early bird is early-described. The late edition is late-described. The unseen object is never-described.

The worm is icky.

```
The high window is in Priority Lab. It is sightline-
described and fixed in place. The initial appearance of
the high window is "There's a tiny high window up near
the ceiling that you can't see unless you're on top of
something."
```

In order for the priorities we just set to be interesting, let's give out some initial appearances and writing a paragraph rules:

```
The initial appearance of the worm is "A worm inches
along the ground."
The initial appearance of the late edition is "Finally, the
late edition lies at your feet."
```

```
Rule for writing a paragraph about the early bird when
the early bird is in a room: say "The early bird always
appears first, and here it is."
Rule for writing a paragraph about the leaf: say "Look,
there's [a leaf][unless the leaf is in the location] on [the
holder of the leaf][end if]!"
Rule for writing a paragraph about an icky thing (called
icky item) which is on something which supports the
player: say "Ew, [an icky item] is right next to you."
```

This procedure also means (as you can test by experiment) that after the late edition has been picked up and dropped again, it lists in no special order in the "you can see..." paragraph (since initial appearances only print when the object has not yet been moved).

```
The afterthought is a thing in the Priority Lab. It is latest-
described.
```

```
The bar stool is an enterable supporter in Priority Lab.
```

```
Test me with "get leaf / drop leaf / look / x unseen object
/ get pebble / look / get twig / look / get afterthought /
look / drop twig / look / get late edition / look / drop late
edition / sit on bar stool / look / get all / put all on stool /
look".
```

Example **Low Light**

RB

An object that is only visible and manipulable when a bright light fixture is on.

Suppose we want a different treatment of lighting than the usual: the room isn't totally dark, but there's something we can't see unless we turn on a bright light.

```
"Low Light"
```

First we make our environment and its light:

```
The Workroom is a room. The desk is in the Workroom.
The brilliant lamp is a device on the desk.


To decide whether the light level is high:
    if the brilliant lamp is switched off, no;
    if the player cannot see the brilliant lamp, no;
    yes.


To decide whether the light level is low:
    if the light level is high, no;
    yes.
```

Now we make a shadow so that the player can only refer to it if the shadow is in inventory or the light is on:

```
The shadow is a privately-named thing on the desk.


Understand "barely-visible" or "barely visible" or
"shadow" as the shadow when the light level is high.
Understand "invisible" or "shadow" as the shadow when
the player encloses the shadow.
```

And finally a couple of extra touches to make it clear why we're able to interact with the shadow when it's in inventory, even if the light is low:

```
Before printing the name of the shadow:
    if the light level is high:
        say "barely-visible ";
    otherwise if the player encloses the shadow:
        say "invisible (but tangible) "
```

```
After dropping the shadow when the light level is low:
    say "You let it go and it fades into the ambient
gloom."
```

To handle the appearance of the object, we want to set its locale priority to 0: that will prevent it being named in room descriptions.

```
After choosing notable locale objects:
    unless the light level is high:
        set locale priority of the shadow to 0.
```

```
Test me with "look / get shadow / turn on lamp / look /
get shadow / i / turn off lamp / i / drop shadow / look /
get shadow / turn on lamp / look".
```

Example **Casino Banale** *RB*

Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

In a work of interactive fiction that involves many new discoveries, we might want to change the way we narrate room descriptions and describe objects as the player learns new information.

One approach to this is to create a model of the facts we want the player to find out, and attach some narrative text to each. When a fact becomes relevant to the story, that narrative text is shown to the player. So:

"Casino Banale"

Section 1 - Procedure

First we create the concept of facts, and the idea that facts can make some things more important than others.

A fact is a kind of thing. A fact can be known or unknown. A fact can be ready to learn or hidden. A fact has some text called the narration.

Definition: a thing is narratively significant if it conveys an interesting fact.

Definition: a thing is narratively dull if it is not narratively significant.

Conveyance relates various things to various facts. The verb to convey means the conveyance relation.

Definition: a fact is interesting if it is unknown and it is ready to learn.

Now, we also need a way to tell Inform to introduce certain new facts when the right previous ones have been introduced. We'll create a "following" relation, according to which a new fact can be told to the player when the player has already learned all the facts it follows. This way, we can simulate the effect of putting together several pieces of evidence to come to a conclusion:

Following relates various facts to various facts. The verb

```
to follow means the following relation.


To say (new fact - a fact):
    say "[narration of the new fact]";
    now the new fact is known;
    repeat with possible outcome running through facts
which follow the new fact:
        if every fact which is followed by possible outcome
is known:
            now the possible outcome is ready to learn.
```

Next we need a way for the game to introduce these new facts. Let's say we want them to come up when the player examines something appropriate, or sees it in the room:

```
After examining something which conveys an interesting
fact (called discovery):
    say "[discovery][paragraph break]".


After choosing notable locale objects:
    repeat through the Table of Locale Priorities:
        if the notable-object entry is narratively significant:
            set the locale priority of the notable-object entry
to 1.


For writing a paragraph about a narratively significant
thing (called item):
    now the item is mentioned;
    let chosen fact be a random interesting fact which is
conveyed by the item;
    say "[chosen fact][paragraph break]".
```

The "after choosing notable locale objects" line here handles things so that any interesting conclusions we want to draw are always given first, followed by the less interesting description.

And finally, we need to give the player a little evidence to piece together:

```
Section 2 - Scenario
```

The Casino is a room.

Frince is a man in the Casino. The description is "Frince is a friend of yours -- if you reckon friendship on the same terms that one reckons a cat as a pet. He spends time with you when he wants to, but if your wishes or convenience ever run counter to a whim of his, it's the whim that wins. Always. [paragraph break]He's also wearing a somewhat ludicrous shirt."

Frince wears a ludicrous shirt. The description of the ludicrous shirt is "Fine white fabric with satiny white pinstripes: it's that expensive, effeminate look that Frince is so fond of, and which -- combined with his name -- gives people completely the wrong idea about him."

Tim is a man in the Casino. The description is "You don't know Tim well. Kind of wall-flowerish. The only thing that seems to excite him is craps."

Penny is a woman in the Casino. The description is "Loud. Brash. Hot, probably, if you can look past the loud and brash."

Rule for writing a paragraph about a narratively dull person:
    let is-are-n be "is";
    if the number of unmentioned narratively dull people is not 1:
        let is-are-n be "are";
    say "[A list of unmentioned narratively dull people] [is-are-n] [one of]watching the croupier[or]following the spin of the roulette[or]chattering[at random][one of] breathlessly[or] impatiently[or][at random]."

Penny-annoying is a fact.
It is ready to learn.
The narration is "[if looking]Penny grimaces at you--[end if]Penny is the same woman who stepped on your toe in the buffet line. The third time, she blurted, 'You have big shoes, don't you?'"
Penny conveys penny-annoying.

lipstick-smudges is a fact.
It is ready to learn.
The narration is "There are a couple of smudges of coral-colored lipstick on the collar."
The ludicrous shirt conveys lipstick-smudges.

penny-wears-coral is a fact.
It follows penny-annoying.
The narration is "[if looking]Penny catches your eye again. [end if]The bright coral lipstick was really not a wise choice."
Penny conveys penny-wears-coral.

Affair-with-penny is a fact.
It follows lipstick-smudges and penny-wears-coral.
The narration is "You avoid [if examining Frince]his[otherwise]Frince's[end if] eye. You need some time to adjust to the image of him making out with Penny in a storage closet before you can talk to him without appalled giggling."
Frince conveys affair-with-Penny.

Test me with "x penny / x frince / x shirt / look".

ExampleKiwi
RB
Creating a raised supporter kind whose contents the player can't see or take from the ground.

Suppose we want there to be some high shelves in our game, which the player can't get at unless he's standing on a prop of some kind. (This is a pretty hoary and over-used puzzle, but there may still be occasions when it becomes useful again.)

In order to resolve this, we want to set up a raised supporter kind. When something is on a raised supporter, it should be mentioned to the player only if the player is in the right position (i.e., standing on something) and otherwise omitted from the description entirely.

```
"Kiwi"

Section 1 - Procedure

A raised supporter is a kind of supporter.

For printing a locale paragraph about a raised supporter
(called the high place):
    if the player is on a supporter (called the riser):
        say "Up on [the high place] (and only visible
because you're on [the riser]) [is-are a list of things on
the high place].";
    otherwise:
        say "The [high place] is above you."
```

Note that here we don't continue the activity because we want to completely replace the normal behavior of describing what is on supporters.

```
Definition: a thing (called target item) is out of reach:
    if the player is on a supporter, no;
    if the target item is on a raised supporter, yes;
    no.
```

Now we also need to prevent the player from interacting with things that are out of reach:

```
Before doing something:
    if the noun is out of reach or the second noun is out of
reach:
        say "You can't reach from down here." instead.
```

...or restoring things to the shelves while the player is in the wrong position...

```
Instead of putting something on a raised supporter when
the player is not on a supporter:
    say "You can't reach from down here."
```

And raised supporters shouldn't be searchable from the ground either:

```
Instead of searching or examining a raised supporter
when the player is not on a supporter:
    say "You can't see from down here."
```

Finally, we need to tackle the case where the player types GET ALL FROM SHELF, because we don't want to list the objects up there if the player can't even see them. We use a rule for deciding whether all includes in order to tell Inform not to consider items that can't be reached, and then we adjust the parser error so that it's a little more instructive than "There are none at all available!", which is what the response would otherwise be:

```
Disallowed-all is a truth state that varies. Disallowed-all
is false.


Rule for deciding whether all includes an out of reach
thing:
    now disallowed-all is true;
    it does not.


Rule for printing a parser error when the latest parser
error is the nothing to do error and the player is not on a
supporter:
    if disallowed-all is true:
        say "Whatever might be up there, you can't see or
```

```
reach from down here.";
    otherwise:
        make no decision.


A first action-processing rule:
    now disallowed-all is false.


Section 2 - Scenario


The Bottom of the Nursery is a room. "Ever since you
ate that mysterious cake, you've been even shorter than
usual."


The high shelf is a raised supporter in the Nursery. It is
scenery. On the high shelf are a kiwi-green ball and a
stuffed dodo.


The step-ladder is an enterable supporter in the
Nursery. Understand "ladder" as the step-ladder.


Test me with "x shelf / search shelf / get dodo / get all
from shelf / stand on ladder / get all from shelf / search
shelf / get off / put all on shelf / get all from shelf / stand
on ladder / put all on shelf".
```

ExampleCopper River *RB*

Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

In a very dense environment, we might want to offer the player room descriptions in which only the currently-interesting items are mentioned,

while other objects are suppressed even if they are present. In effect, this takes the idea of scenery and makes it more flexible: different things might become background objects or foreground objects at different times during play.

There are a wide range of possible reasons to do this -- to shift the narrative emphasis, to change the mood of the game by highlighting different parts of the environment, to show the game from the perspective of different viewpoint characters -- but in the following example, our goal is to show the player only the objects that are currently useful for puzzles.

To do this, we need some notion of what puzzles are currently available and unsolved, so we make an "unsolved" adjective; we also need to know which things solve the puzzle, so we create a "resolving" relation, to indicate which objects resolve which problems.

Given that information, we can create rules about which objects in the game world are currently interesting, which are currently dull, and describe accordingly:

```
"Copper River"

Use scoring.

Section 1 - Procedure

Resolving relates various things to various things. The
verb to resolve means the resolving relation.

Definition: a thing is interesting if it is not dull.

Definition: a person is dull:
    no.
```

```
Definition: a thing is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    yes.


Definition: a supporter is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    if it supports an interesting thing, no;
    yes.


Definition: a container is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    if it contains an interesting thing, no;
    yes.


After choosing notable locale objects:
    repeat with item running through unsolved things:
        set the locale priority of the item to 1.


For printing a locale paragraph about a dull thing (called
item):
    now the item is mentioned.


Before printing a locale paragraph about a supporter
(called item):
    now every dull thing on the item is mentioned.


Before printing a locale paragraph about a container
(called item):
    now every dull thing on the item is mentioned.


Instead of searching a supporter:
```

```
    if the noun supports something interesting:
        say "[A list of interesting things on the noun] [are]
on [the noun]";
        if the noun supports something dull:
            say " (alongside [a list of dull things on the
noun])";
        say ".";
    otherwise if the noun supports something dull:
        say "There's nothing very useful here, only [a list of
dull things on the noun].";
    otherwise:
        say "[The noun] [are] completely bare."
```

```
Instead of searching a container:
    if the noun contains something interesting:
        say "[A list of interesting things in the noun] [are] in
[the noun]";
        if the noun contains something dull:
            say " (alongside [a list of dull things in the
noun])";
        say ".";
    otherwise if the noun contains something dull:
        say "There's nothing very useful here, only [a list of
dull things in the noun].";
    otherwise:
        say "[The noun] [are] completely empty."
```

```
Before listing contents when not taking inventory: group
dull things together.
```

```
Rule for grouping together dull things: say "assorted dull
items".
```

Section 2 - Scenario World and Objects

```
The Kitchen is a room. "Your Aunt Fiona's kitchen looks
as though it has been at the eye of a glitter storm. Fine,
sparkling grit dusts every surface. The appliances are
slightly askew, too, as though they hadn't quite settled
```

after a vigorous earthquake."

The shelf is a scenery supporter in the Kitchen. On the shelf is a can of beans, a can of potato leek soup, and a tin of deflating powder.

The cabinet is a scenery container in the Kitchen. In the cabinet is a book of matches, a bottle of descaling solution, a fish hook, and a rusty knife. It is openable and closed.

The counter is a scenery supporter in the Kitchen. On the counter is an espresso machine, a blender, and a mortar. The blender and the mortar are containers. In the mortar is a pestle. Understand "countertop" as the counter.

The stove is a scenery supporter in the Kitchen. The oven is part of the stove. The oven is a closed openable container.

The refrigerator is a fixed in place container in the Kitchen.

Understand "fridge" as the refrigerator.

The description is "The refrigerator is a dull blue-green, and has a puffy, marshmallow texture on the outside, which means that it's no good for sticking magnets to. Aunt Fiona has never been willing to explain where she got it." The refrigerator is openable and closed.

In the refrigerator are a bottle of ice wine, a bag of carrot sticks, and an egg.

Aunt Fiona is a woman in the Kitchen. Aunt Fiona can be inflated or deflated. Aunt Fiona is inflated. "[if Aunt Fiona is inflated]Aunt Fiona stands nearby. Or perhaps 'stands' is the wrong word: she has been sort of puffed up in her own skin like a balloon, and is now propped in a corner of the room with her head lolling back[otherwise]Aunt Fiona stands -- on her own two slender legs -- at the center of the room[end if]."

Every turn when Fiona is unsolved and Fiona can see the player:
    if a random chance of 1 in 3 succeeds:
        say "[one of]Aunt Fiona's eyes follow you, wide and desperate, but it doesn't look like she's able to do anything[or]Aunt Fiona is still looking reproachful[or]A faint gurgling comes from Aunt Fiona[or]Aunt Fiona makes a funny croak noise[or]Aunt Fiona is still having trouble speaking. Perhaps her throat is as swollen as the rest of her[or]Aunt Fiona twitches[stopping]."

There is a thing called a salmon. Understand "fish" as the salmon. The salmon can be scaly or prepared. The salmon is scaly. The description is "[if scaly]It looks delicious, but is still covered with scales[otherwise]The salmon has been scaled and is ready to eat[end if]."

Before printing the name of the salmon when the salmon is scaly:
    say "very scaly ".

Section 3 - Scenario Puzzles

Definition: Aunt Fiona is unsolved if she is inflated.
Definition: the salmon is unsolved:
    if the salmon is off-stage, no;
    if the salmon is scaly, yes;
    no.

The deflating powder resolves Aunt Fiona.

Instead of putting the deflating powder on Aunt Fiona:
    try throwing the deflating powder at Aunt Fiona.
Instead of giving the deflating powder to Aunt Fiona:
    try throwing the deflating powder at Aunt Fiona.

Instead of throwing the deflating powder at Aunt Fiona:
    if Aunt Fiona is inflated:
        say "You toss some of the powder in Aunt Fiona's direction, and with a sudden gaseous HUFF! she returns to her usual shape and size. [paragraph break]'Well!' she says, brushing herself off. 'That was bracing!' [paragraph break]You give her an embarrassed smile, to apologize for not curing her faster.";
        now Aunt Fiona is deflated;
        increase the score by 2;
    otherwise:
        say "[one of]You throw another hefty dose of the powder at your aunt. [paragraph break]'Thank you, child,' she says, sneezing. 'But I think you've done enough now.'[or]You throw another hefty dose of the powder at your aunt. [paragraph break]'You're too kind,' she wheezes, through a cloud of glittering dust. [or]You've probably done enough with the powder. [stopping]".

Every turn when Aunt Fiona is deflated and the salmon is off-stage:
    move the salmon to the counter;
    say "'At least they didn't get this,' she says, producing from somewhere on her person a fresh-caught salmon. An odd pattern around its eye sockets makes it looks comically as though it wears spectacles. 'It's the Salmon of Knowledge,' she explains casually. 'We just need to scale and cook it.'"

The bottle of descaling solution resolves the salmon.

```
Does the player mean putting the descaling solution on
the fish hook: it is unlikely.
Does the player mean putting the descaling solution on
the salmon: it is very likely.
```

```
Instead of putting the bottle of descaling solution on the
salmon:
    if the salmon is scaly:
        now the salmon is prepared;
        say "With just a single squirt of the descaling
solution (which confusingly has a picture of bathroom
tiles on the label), you remove the scales from the
salmon, leaving its pink flesh ready for preparation.";
        increase the score by 2;
    otherwise:
        say "'Don't do that,' Aunt Fiona warns you.
'Excessive applications could damage the flesh.'"
```

```
Test me with "look / get powder / drop powder / look /
look in cabinet / get powder / put powder on fiona / look
/ open cabinet / look in cabinet / get solution / open
fridge / put solution in fridge / look / get solution / put
solution on salmon / look".
```

ExamplePeeled

RB

Two different approaches to adjusting what the player can interact with, compared.

Suppose we have a situation where the player is in darkness, but is allowed to feel and interact with (except for examining) any large objects. In that case, we write a scope rule that puts those large objects into scope all the time, and trust the "requires light" aspect of verbs like examining to prevent the player from doing any actions that he shouldn't:

```
"Peeled"
```

```
A thing can be large or small.
```

```
Before touching a large thing when in darkness: say
"You grope for [the noun]..."


After deciding the scope of the player:
    repeat with item running through large things in the
location:
        place item in scope.


Some generic surroundings are backdrop. They are
everywhere. Understand "walls" or "wall" or "ceiling" or
"ground" or "floor" or "area" or "room" or "here" as the
generic surroundings. Instead of touching the generic
surroundings: say "You encounter nothing
extraordinary." Instead of touching the generic
surroundings when in darkness: say "You try feeling
your way around and reach [a list of large things in the
location]." After deciding the scope of the player when in
darkness: place the surroundings in scope.


The Room of Mystery is a dark room. The bearskin rug
is a large thing in the Room of Mystery. Instead of
touching the rug: say "It feels furry!"


The peeled grape is a small thing in the Room of
Mystery. Instead of touching the peeled grape: say
"Gosh, is that an eyeball?"


Test me with "feel floor / feel rug / eat rug / examine rug
/ get grape".
```

Sadly, because the grape is small, the player will never encounter this horror.

Alternatively, suppose we have a situation in which the player can use one command to interact with a kind of thing that isn't normally in scope. It's usually most convenient to write the "understand" rule appropriately

rather than use the scope activity.

(Note that we define "inquiring about" as applying to one *visible* thing; otherwise we would be required to be able to touch the catsuit in order to inquire about it. More on this restriction may be found in the Advanced Actions chapter on the topic of visible, touchable, and carried things.)

```
"Peeled"

Mr Steed's Flat is a room.

Understand "ask about [any subject]" as inquiring about.
A subject is a kind of thing. The skintight catsuit is a
subject. Inquiring about is an action applying to one
visible thing.

Carry out inquiring about something:
    say "'What can you tell me about [the noun]?' you
demand. Mr Steed raises his eyebrows, but does not
reply."

Test me with "ask about catsuit / x catsuit".
```

All this said, there do arise certain complex situations when we want an activity-specific scoping.

ExampleFour Stars 2 *RB*

Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

As we have seen, a well-written understand rule will often solve the problem of allowing the player to apply specific actions to objects not normally in scope. When we need to adjust scope for some other reason than reading the player's command, though, "deciding the scope of..."

may come in handy.

For instance, suppose we wanted to extend Four Stars 1 to add a tomcat on the balcony that will be heard whenever the player listens from the next room, as in:

```
>listen
You hear the soothing whalesong from the Bose
speaker and the yowling from the tomcat.
```

To do this, we need to make sure that in the rule that assembles our listening description,

```
Instead of listening to a room:
    if an audible thing can be touched by the player, say
"You hear [the list of audible things which can be
touched by the player].";
    otherwise say "A merciful peace prevails."
```

now includes the tomcat in the "list of audible things which can be touched by the player".

To this end, we're going to change the way we assess scope, but only during the listening action. Otherwise the tomcat remains in the other room and off-limits. The new source text is marked out below:

```
"Four Stars"


Section 1 - Procedure


A thing has some text called sound. The sound of a
thing is usually "silence".


The report listening rule is not listed in the report
listening to rules.
```

Carry out listening to something:
    say "From [the noun] you hear [the sound of the noun]."

Instead of listening to a room:
    if an audible thing can be touched by the player, say "You hear [the list of audible things which can be touched by the player].";
    otherwise say "A merciful peace prevails."

Definition: a thing is audible if the sound of it is not "silence".

Before printing the name of something audible while listening to a room:
    say "[sound] from the "

A thing has some text called scent. The scent of a thing is usually "nothing".

The report smelling rule is not listed in the report smelling rulebook.

Carry out smelling something:
    say "From [the noun] you smell [scent of the noun]."

Instead of smelling a room:
    if a scented thing can be touched by the player, say "You smell [the list of scented things which can be touched by the player].";
    otherwise say "The place is blissfully odorless."

```
Definition: a thing is scented if the scent of it is not
"nothing".


Before printing the name of something scented while
smelling a room: say "[scent] from the "
```

Here is our addition:

```
After deciding the scope of the player while listening or
sleeping or looking:
    if in darkness:
        repeat with locale running through adjacent rooms:
            place locale in scope.


A reaching inside rule while listening or sleeping or
looking:
    rule succeeds.


Section 2 - Scenario


The Waning Moon Resort is a dark room. "A spacious
room with a flagstone floor, and a dreamcatcher hung
over the king-size bed." The dreamcatcher is scenery in
the Resort. The description is "The usual web of threads
and crystals, feathers and beads." Instead of taking the
dreamcatcher, say "Ah, ah -- you might be tempted to
take it as a souvenir, except that the price list in the
minibar clearly states they charge $65 apiece if you
walk off with one. Cheaper than stealing the Frette
bathrobes, but still probably not a good idea."
```

And now our threat to the player's peace:

```
The Balcony is outside from the Resort. In the Balcony
```

is a tomcat. The sound of the tomcat is "yowling". After printing the name of the tomcat when the tomcat is not visible: say " outside on the balcony".

From here we continue with the same scenario as before:

The king-size bed is an enterable supporter in the Resort. The description is "200-thread-count Egyptian cotton sheets, according to the website. You would make fun, only they really are extraordinarily comfortable." The player is on the bed. A Lindt chocolate is on the bed. It is edible. The scent of the chocolate is "chocolate-hazelnut smell".

Instead of exiting: say "You are too weary to move."

The suitcase is an openable closed container in the Resort.

An electric light is a kind of device. Carry out switching on an electric light: now the noun is lit. Carry out switching off an electric light: now the noun is unlit. Understand "light" as an electric light.

The solar lamp is an electric light in Waning Moon Resort. The description is "Specially designed to give light in a spectrum resembling sunlight, to improve the mood and make a person energetic." The lamp is switched on and lit.

An electric noisemaker is a kind of device. An electric noisemaker has some text called usual sound. The usual sound of an electric noisemaker is usually "beepbeepbeep". Carry out switching on an electric noisemaker: now the sound of the noun is the usual sound of the noun. Report switching on an electric noisemaker: say "[The noun] goes [usual sound of the

noun]!" instead. Report switching off an electric noisemaker: say "You switch off [the noun], silencing the [usual sound of the noun]." instead.

Carry out switching off an electric noisemaker: now the sound of the noun is "silence".

The bedside table is in the Resort. The table supports a potted plant and a Bose speaker. The scent of the potted plant is "rosemary"

The Bose speaker is an electric noisemaker. The usual sound of the speaker is "soothing whalesong". The sound of the speaker is "soothing whalesong". The speaker is switched on.

Instead of touching a device: say "You feel the surface of [the noun] and discover the switch."

Instead of touching a scented thing: say "The brush of your fingers stirs loose a fresh cloud of [scent of the noun] smell."

Rule for printing the description of a dark room: try listening; try smelling; rule succeeds.

Instead of examining an audible thing while in darkness: try listening to the noun. Instead of examining something while in darkness: try touching the noun.

Before touching something when in darkness:
    say "You grope about..."

After inserting the plant into something:
    say "You unceremoniously dump [the noun] into [the second noun], hoping it sustains no important damage thereby."

Before printing the name of a dark room: if the player can touch an audible thing, say "Noisy "; if the player can touch a scented thing, say "Perfumed ".

Visibility rule when in darkness:
    if examining something, there is sufficient light;
    there is insufficient light.

Rule for printing the announcement of darkness: say "It is now pleasantly lightless in here." instead.

Rule for deciding the scope of the player while in darkness: place the location in scope.

To decide whether in daylight:
    if in darkness, no;
    yes.

Instead of sleeping when in daylight:
    say "You've never been able to sleep with the light on."

Instead of sleeping when the player can touch an audible thing (called the irritant):
    say "The steady [sound of the irritant] from [the irritant] prevents your slumber."

```
Instead of sleeping when the player can touch a scented
thing (called the irritant):
    if the irritant is chocolate, say "The smell of chocolate
continues to tantalize you, keeping you from sleep.";
    otherwise say "You sniffle. [The irritant] is probably
acting on your allergies."


Instead of sleeping:
    say "You slip easily into the arms of Morpheus.";
    end the story finally saying "At last..."


When play begins:
    say "You have at last escaped from the airport and
gotten through customs; survived an unnerving taxi ride
over icy highways; stared down the impertinent
concierge; endured the bellhop's catalog of features in
your room; and achieved, finally, a moment of peace.
Time for a good night's slumber!"


Test me with "listen / x dreamcatcher / switch lamp off /
look / sleep / eat chocolate / sleep / get plant / examine
plant / open suitcase / put plant in suitcase / close
suitcase / sleep / look / examine bose / switch bose off /
sleep".
```

Of course, this new version is less happy for the player, as we haven't included any way to silence the cat.

ExampleGinger Beer *RB*

A portable magic telescope which allows the player to view items in another room of his choice.

Suppose we want to have a pair of linked lenses so that the player can look into one of them and see things which occur in room containing the other lense.

We begin simply with a bit of environment for the player to wander around:

"Ginger Beer"

The Ginger Beer Factory is a room. "In the center of the room is an enormous pot filled with crushed ginger, which seems to be bubbling slightly on its own. The fumes are overwhelming."

The pot is scenery in the Ginger Beer Factory. The description of the pot is "Cast iron." In the pot is a bubbling brew.

Instead of smelling the Ginger Beer Factory: try smelling the brew.

Instead of smelling the brew, say "You blink back tears."

The Storeroom is south of the Ginger Beer Factory. "The walls here are lined with a prodigious number of small, rounded bottles, each with a screw top and a smiling pirate on the label."

The Clippings Room is west of the Ginger Beer Factory. "A clean room lined with steel tables, for preparing ingredients."

Some steel tables are a supporter in the Clippings Room. They are scenery. The description is "They are roughly the size and height of laboratory worksurfaces."

The quantity of dandelion is on the steel tables. The description is "Horrible common weed."

The wooden box is on the steel tables. It is openable and closed. The description is "A large wooden box with a lid, used for ingredient storage. There is a label on the lid."

The label is part of the box. The description is "BURDOCK: the root beaten with a little salt and laid on the place suddenly easeth the pain thereof, and helpeth those that are bit by a mad dog:... the seed being drunk in wine 40 days together doth wonderfully help the sciatica: the leaves bruised with the white of an egg and applied to any place burnt with fire, taketh out the fire, gives sudden ease and heals it up afterwards.... The root may be preserved with sugar for consumption, stone and the lax."

The quantity of burdock is in the box. The description is "It looks like a kind of thistle."

Some bottles are in the Storeroom. They are scenery. The description is "They are smaller than the average bottle, because more potent." Instead of taking the bottles, say "Take one away and the whole lineup will cascade to the floor."

Now for the lenses themselves:

A lense is a kind of thing.

The large end of the telescope is a lense in the Ginger Beer Factory. "There is a large glass lense propped

```
against the wall, in which are reflected all the contents
of the room." Understand "glass" or "lense" as the large
end.
```

```
The small end of the telescope is a lense in the
Storeroom. "There is a small glass lense sitting on the
floor. Due to some curious effect of the optics, it
appears to be giving a view of somewhere else entirely."
Understand "glass" or "lense" as the small end. The
description is "A gleaming lense about the size of a
pound coin."
```

Here is the critical bit, which needs to be somewhat flexible, since the large end of the telescope could in theory be left anywhere in the game (and should still work).

```
After deciding the scope of the player while the small
end is carried by the player:
    let there be the holder of the large end;
    place there in scope.
```

```
Before searching the small end when the small end is
not carried by the player:
    say "(first picking up [the small end] and holding it to
your eye)";
    silently try taking the small end.
```

```
Instead of searching the small end when the player is
not carrying the small end:
    say "It's too hard to look through the small end from a
distance."
```

```
Instead of searching the large end,
    say "You see only your own reflection."
```

We also want to make sure that the player who looks through the small lense does not see the large lense listed among the contents of the other location:

Definition: a thing is recognizable if it is not a lense.

Instead of searching the small end:
    let the far side be the holder of the large end of the telescope;
    say "You peer into the little lense and through it see, in [the far side], [the list of recognizable things in the far side]."

Test me with "examine lense / south / examine lense / look through lense / north / look through small lense".

And we're done.

---

366 **Example** **Rock Garden** *RB*

A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

A map of linked rooms works well for modeling enclosed or indoor space, and somewhat less well for modeling large open spaces, where a person should reasonably be able to see things which are much too far away to touch. With some modifications to scoping, though, we can create an environment where objects in nearby rooms are described and viewable, and where the player will automatically move towards distant items before interacting with them physically.

"Rock Garden"

Section 1 - General Rules

Intervisibility relates rooms to each other in groups. The

verb to be connected with means the intervisibility relation.

Definition: a room is inter-visible if it is connected with more than one room.

After deciding the scope of the player when the location is an inter-visible room:
    repeat with other place running through rooms which are connected with the location:
        unless the other place is the location, place the other place in scope.

Rule for reaching inside a room (called target) which is connected with the location:
    let way be the best route from the location to the target;
    if the way is not a direction:
        say "You can't get over to [the target] from here.";
        deny access;
    say "(first heading [way])[command clarification break]";
    try going way;
    if the player is in the target, allow access;
    otherwise deny access.

After looking when the location is an inter-visible room:
    repeat with other place running through rooms which are connected with the location:
        if the other place is not the location, describe locale for other place.

Section 2 - The Scenario

Rock Garden West is west of Rock Garden East. Rock Garden East contains a rake. Rock Garden West

contains a bench and a maple leaf. The bench is an enterable supporter.

Rock Garden West is connected with Rock Garden East.

Test me with "get rake / drop rake / sit on bench / get rake".

ExampleStately Gardens *RB*

An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

This time we're going to assume that the player can see into any room that is on a line of sight within one or two steps of travel.

"Stately Gardens"

Chapter 1 - Laying Out Rooms

A room can be indoors or outdoors.

Use full-length room descriptions.

After deciding the scope of the player:
    repeat with the way running through directions:
        let first step be the room the way from the location;
        if the first step is a room:

```
        place the first step in scope;
        let second step be the room the way from the
first step;
        if the second step is a room, place the second
step in scope;
    place the obelisk in scope.
```

The obelisk is so large that it can be seen from every room. If we had a number of such large monuments we might want to write a systematic routine to handle them, but this will do for now.

```
The room description heading rule is not listed in the
carry out looking rules.
```

Now, we set things up so that the surrounding areas are described automatically as part of the room description:

```
Building description is a truth state that varies. Building
description is false.


After looking when the location is an outdoors room:
    now count of sentences is 0;
    now building description is true;
    repeat with way running through directions:
        let space be the room way from the location;
        if space is an outdoors room, silently try looking
toward space;
    if the obelisk is not in the location and the obelisk is
unmentioned:
        let the way be the best route from location to the
Upper Terrace;
        if the way is a direction, say "[The obelisk] is
proudly visible on [the way] horizon. [run paragraph
on]";
        increment the count of sentences;
    now building description is false;
    unless the count of sentences is 0:
        say paragraph break.
```

But perhaps there are a few rooms where we do not wish that to happen, so we'll build in exceptions for those.

```
After looking in the rose garden:
    say "Otherwise, you are quite cut off.".


After looking in the Ha-ha:
    do nothing.
```

And suppose we want to allow the player to look in any direction:

```
Understand "look [direction]" or "look to/toward
[direction]" as facing.


Facing is an action applying to one visible thing.


Carry out facing:
    let the viewed item be the room noun from the
location;
    if the viewed item is not a room:
        if the location is indoors, say "Your view is
restricted by the lack of doors or windows in that
direction." instead;
        otherwise say "You can't see anything promising
that way." instead;
    try looking toward the viewed item.


Instead of facing up:
    say "Above you is bright sky."
```

We also need to tell distant rooms how to describe themselves.

```
Understand "look toward [any adjacent room]" as
looking toward.


Looking toward is an action applying to one visible thing.
```

```
Check looking toward a room which does not contain
something mentionable:
    if building description is false:
        say "You can't make out anything of interest that
way." instead.



Carry out looking toward:
    now every thing is unmentioned;
    now the chosen direction is the best route from the
location to the noun;
    now the second noun is the room the chosen
direction from the noun;
    if the noun contains something mentionable:
        repeat with item running through mentionable
things in the noun:
            carry out the writing a distant paragraph about
activity with the item;
    if the noun contains something mentionable:
        increment the count of sentences;
        choose row count of sentences in the Table of
Distance Sentences;
        if the second noun is an outdoors room and the
second noun contains something mentionable, say
"[both entry] [run paragraph on]";
        otherwise say "[here entry] [run paragraph on]";
    otherwise:
        if the second noun is an outdoors room and the
second noun contains something mentionable:
            increment the count of sentences;
            choose row count of sentences in the Table of
Distance Sentences;
            say "[there entry] [run paragraph on]";
    if building description is false:
        say paragraph break.
```

And again, some exception needs to be made for seeing what's in the dip in the ground:

```
Instead of looking toward the Ha-ha:
    now the chosen direction is the best route from the
location to the noun;
    now the second noun is the room the chosen
direction from the noun;
    if the second noun is an outdoors room and the
```

```
second noun contains something mentionable:
    increment the count of sentences;
    choose row count of sentences in the Table of
Distance Sentences;
    say "[there entry] [run paragraph on]".
```

The following is to account for cases where the player types "look toward obelisk" or similar, rather than looking toward a room:

```
Understand "look toward [something]" as examining.
```

The following is arguably an unnecessary refinement, but the listing of items in the distance gets a bit repetitive unless we vary the sentence structure.

```
Chosen direction is a direction that varies.
```

```
Count of sentences is a number that varies.
```

```
Table of Distance Sentences
```

there	"From here, you make out [a list of mentionable things in the noun]."
[one of]	"To the [chosen direction], there [is-are a list of mentionable things in the second noun], partly obscuring your further view of [a list of mentionable things in the second noun]."
[one of]	"Meanwhile, to the [chosen direction] is a list of mentionable things in the second noun]."
[one of]	"When you [choose direction], you see [a list of mentionable things in the second noun], and somewhat further on [a list of mentionable things in the second noun]."
[one of]	"Meanwhile, to the [chosen direction] [is-are a list of mentionable things in the second noun], which, [a list of mentionable things in the second noun]."
[one of]	"Meanwhile, to the [chosen direction] [is-are a list of mentionable things in the second noun]."
[one of]	"Meanwhile, to the [chosen direction] is a list of mentionable things in the second noun]."

noun]."

"Finally, [one or other] [is] [visible] if [the first noun] is [visible] and [the second noun] is [visible] and [the first noun] is in the second noun [or the second noun is a list of mentionable things and the first noun is one of the things in the second noun]."

Now, our ability to view things at a distance should be determined by the size of the things we're trying to see:

Chapter 2 - Height

A height is a kind of value. 10 feet 11 inches specifies a height. 10 feet 11 specifies a height. The verb to stand means the height property. The verb to measure means the height property. A thing has a height. The height of a thing is usually 3 feet 0.

Definition: a thing is tiny if its height is 0 feet 6 inches or less.

Definition: a thing is short if its height is 3 feet 0 or less.

Definition: a thing is tall if its height is 6 feet 0 or more.

The height of a man is usually 5 feet 10 inches. The height of a woman is usually 5 feet 6 inches.

Definition: a thing is monumental if it is taller than 25 feet 0 inches.

Definition: a thing is mentionable if it stands tall enough

```
to see.


To decide whether (item - a thing) stands tall enough to
see:
    if the item is in the Rose Garden and the item is
shorter than the roses, no;
    if the item is mentioned, no;
    if the item is in an adjacent room and item is taller
than 2 feet 0, yes;
    if the item is taller than 4 feet 0, yes;
    no.


Instead of examining something which is within a room
(called the space) which is not the location:
    if the location is adjacent to the space:
        if the noun is tiny, say "It is too far from here for
you to make out much detail about [the noun]." instead;
        let way be the best route from the location to the
space;
        if the way is a direction, say "You gaze off [way] at
[the noun]...";
        continue the action;
    otherwise:
        if the noun is short, say "It is too far from here for
you to make out much detail about [the noun]." instead;
        let way be the best route from the location to the
space;
        if the way is a direction, say "You gaze off [way]
into the distance at [the noun]...";
        continue the action.
```

We might also want to be able to override, manually, the way distant things are described.

```
Writing a distant paragraph about something is an
activity.


Rule for writing a distant paragraph about the lily pond:
    if the second noun is a room and something
mentionable is in the second noun, say "A [lily pond],
[chosen direction], patchily reflects [a list of mentionable
```

```
things in the second noun] on the far side. [run
paragraph on]";
    otherwise say "To [the chosen direction], [a lily pond]
shimmers in the sunlight. [run paragraph on]"
```

```
Rule for writing a distant paragraph about the roses:
    if something in the Rose Garden is taller than the
roses,
        say "Over the tops of [the roses], [chosen
direction], you see [a list of mentionable things in the
rose garden]. [run paragraph on]";
    otherwise say "Immediately [chosen direction] is [the
roses]. [run paragraph on]"
```

```
Rule for writing a distant paragraph about the obelisk:
    if a mentionable thing in the Upper Terrace is shorter
than the obelisk,
        say "A stupidly grand [obelisk], [chosen direction],
towers over [a list of mentionable things in the Upper
Terrace]. [run paragraph on]";
    otherwise say "To [the chosen direction], you can't
help noticing [the obelisk], which is much larger than
any object really needs to be. [run paragraph on]".
```

```
After writing a distant paragraph about something:
    increment the count of sentences.
```

Moreover, proximate things might have special descriptions too.

```
Rule for writing a paragraph about something tiny when
the location is outdoors:
    if the location is the Gravel Circle,
        say "Abandoned in the gravel [is-are a list of
unmentioned tiny things in the location]. [run paragraph
on]";
    otherwise say "Half trampled into the grass, and easy
to miss, [is-are a list of unmentioned tiny things in the
location]. [run paragraph on]"
```

```
Before doing something other than examining or
approaching to something which is not within the
location:
    if the player has the noun, continue the action;
    say "(first going over to [the noun])[line break]";
    try approaching the noun;
    if the noun is not within the location, stop the action.


Understand "go toward/to/towards/near [something]" or
"approach [something]" as approaching.


Approaching is an action applying to one visible thing.


Check approaching:
    if the player is in something, say "You'll have to get
up." instead;
    if the noun is within the location, say "You're as close
to [the noun] as you can get." instead;
    let space be the location of the noun;
    if the space is not a room, say "You don't quite see
how to get there." instead;
    let way be the best route from the location to the
space;
    unless way is a direction,
        say "You can't see how to get over there from
here." instead.


To head to (space - a room):
    let the way be the best route from the location to the
space;
    if the space is adjacent to the location,
        try going way;
    otherwise silently try going way.


Carry out approaching:
    let space be the location of the noun;
    while the space is not the location:
        head to space.
```

This is a bit primitive, since if we had an occasion where going was blocked, we could get stuck in a loop. So we would need to be careful, but for this example it won't arise.

```
Going state is a truth state that varies. Going state is
false.


Check going:
    now going state is true.


The description of a room is usually "[if going state is
true]You drift [noun] across the open lawn[direction
relative to obelisk]. [end if]An absolutely phenomenal
quantity of manicured turf stretches from where you
stand in almost every direction."


Before reading a command:
    now going state is false.


To say direction relative to obelisk:
    if obelisk is in the location:
        say ", as though drawn magnetically to the foot of
the monument";
    otherwise:
        let way be the best route from the location to the
Upper Terrace;
        if way is the noun, say ", drawn towards [the
obelisk]";
        if the way is the opposite of the noun, say ",
keeping [the obelisk] more or less at your back".


Chapter 3 - The Grounds


When play begins:
```

now the left hand status line is "Idyllic";
now the right hand status line is " ".

The Gravel Circle, the Ha-ha, the Sheep Field, the Open Lawn, the Croquet Ground, the Rose Garden, the Upper Terrace, the Middle Terrace, and the Lower Terrace are outdoors.

The Middle Terrace is north of the Lower Terrace and south of the Upper Terrace. The lily pond is fixed in place in the Middle Terrace."You [if going state is true]come to[otherwise]are at[end if] the north edge of a perfectly round lily pond, bordered with stones. Its surface patchily reflects [the marble anteater] on the south bank." A tent peg and a wilted orchid are in the Middle Terrace. The tent peg measures 0 feet 6. The orchid measures 0 feet 4.

The description of the Lower Terrace is "[if going state is true]You climb [noun] up a small hillock[direction relative to obelisk][otherwise]You stand on a short, round, entirely artificial hillock[end if]."

The marble anteater is a fixed in place thing in the Lower Terrace. The height of marble anteater is 6 feet 2 inches."A marble anteater stands on a pedestal at the top of the hill. In the bright sunlight the white marble makes a striking contrast with [the obelisk] in the distance." The description is "The anteater is very much more than life-size."

The obelisk of black granite is a fixed in place thing in the Upper Terrace."Now that you are at the foot of it, you can properly appreciate the stupid immensity of the obelisk, pointing stonily at heaven." The height of the obelisk is 50 feet 0 inches. The description of the obelisk is "It stands ridiculously tall, and has an inscription on the face."

The inscription is part of the obelisk. The height of the inscription is 0 feet 3 inches. The description of the inscription is "You can't read the squirming, pointed letters, but they make you uneasy.".

The Gravel Circle is west of the Upper Terrace, northwest of the Middle Terrace, and north of the Croquet Ground. The description of the Gravel Circle is "[if going state is true]You head [noun] until the lawn thins and[otherwise]Here the lawn[end if] gives way to a circle of raked gravel, which crunches pleasingly beneath you."

Instead of going northwest in the Upper Terrace, try going north.

The Ha-ha is north of the Gravel Circle and northwest of the Upper Terrace. The description of the Ha-ha is "[if going state is true]The land dips here so suddenly that you do not know the dip is there until you're in it; but it prevents livestock from crossing barriers, and that is the important thing[otherwise]You are at the base of a steep-sided depression, so the lawn continues north and south more or less at the level of your head[end if].

The tip of [the obelisk] is the only thing you can make out from this depression, off to the southeast.". North of the Ha-ha is the Sheep Field. In the Sheep Field is an animal called a black sheep. The black sheep stands 4 feet 3 inches."A black sheep grazes placidly nearby." The description of the black sheep is "It reminds you of your Uncle Tim."

Before going from the Ha-ha:
    say "It's a bit of a scramble to get back up the side of the depression, and you keep slipping in the damp grass. But you manage at last."

The Rose Garden is southwest of the Lower Terrace. The thicket of red roses is a fixed in place thing in the Rose Garden. The thicket stands 4 feet 2 inches."Heavy red roses grow over a roughly horseshoe-shaped wall around you. Over this barrier, the head of [the marble anteater] is visible to the northwest, and the tip of [the obelisk] in the distance."

The description of the Rose Garden is "[if going state is true]You slip [noun] into the enclosure of the rose garden. [end if]The rest of the park, and the world, seems muted and quiet."

Instead of smelling the rose garden: try smelling the roses. Instead of smelling the roses, say "The smell tickles the back of your throat and makes you want to cough."

Instead of listening to the rose garden:
    say "You can't hear anything at all."

The Open Lawn is north of the Rose Garden, west of the Lower Terrace, and southwest of the Middle Terrace. The Croquet Ground is north of the Open Lawn, west of the Middle Terrace, southwest of the Upper Terrace, and northwest of the Lower Terrace.

A discarded champagne cork is in the Open Lawn. It stands 0 feet 2 inches.

A stone bench is an enterable supporter in the Croquet Ground. It stands 3 feet 8 inches."There is a stone bench here -- a sort of stone sofa, really, with nymphs disporting themselves on the arms and back." The

description of the bench is "It used to be a Roman sarcophagus -- hence the nymphs -- but someone has thoughtfully recarved it as lawn furniture."

The half-size Bentley is a vehicle in the Gravel Circle."A sort of child's-toy version of a Bentley is parked [if something parkable is in the location]beside [the tallest parkable thing in the location][otherwise]close at hand[end if]." The description of the half-size Bentley is "Of beautiful and unambiguously luxurious lines, but sized down to hold only one or (at a stretch) two people, and powered by electricity." The half-size Bentley stands 3 feet 6 inches.

Definition: a thing is parkable if it is not a person and it is not the Bentley.

Instead of touching the obelisk, say "Though it is black stone in sunlight, the obelisk is very cold to the touch."

Test me with "look east / look toward obelisk / s / s / e / sw / ne / n / n / w / n / n / examine obelisk / touch obelisk / read inscription".

ExampleApples

RB

Prompting the player on how to disambiguate otherwise similar objects.

Inform by default detects whether two objects can be disambiguated by any vocabulary available to the player. If so, it asks a question; if not, it picks one of the identical objects at random.

Generally this produces good behavior. Occasionally, though, two objects have some distinguishing characteristic that doesn't appear in the object name. For instance, suppose we've created a class of apples that can be told apart depending on whether they've been bitten or not:

```
An apple is a kind of thing. Consumption is a kind of
value. The consumptions are pristine and bitten. An
apple has a consumption. The description of an apple is
"It is [consumption]."
```

```
Understand the consumption property as describing an
apple.
```

The player can meaningfully type

```
>EAT BITTEN APPLE
```

or

```
>EAT PRISTINE APPLE
```

but if he types

```
>EAT APPLE
```

Inform will, annoyingly, ask

```
Which do you mean, an apple or the apple?
```

This gives the player no indication of why Inform is making a distinction. So here we add a special "printing the name" rule to get around that situation:

```
"Apples"
```

```
Orchard is a room.
```

```
An apple is a kind of thing. Consumption is a kind of
value. The consumptions are pristine and bitten. An
apple has a consumption. The description of an apple is
"It is [consumption]."


Understand the consumption property as describing an
apple.


Before printing the name of an apple while asking which
do you mean: say "[consumption] ". Before printing the
plural name of an apple while asking which do you
mean: say "[consumption] ".


The player carries three apples.


Instead of eating a pristine apple (called the fruit):
    say "You take a satisfying bite.";
    now the fruit is bitten.


Instead of eating a bitten apple (called the fruit):
    say "You consume the apple entirely.";
    now the fruit is nowhere.
```

Inform will also separate the bitten from the pristine apples in inventory listings and room descriptions, even though it's not clear why; we can improve on that behavior thus:

```
Before listing contents: group apples together.


Rule for grouping together an apple (called target):
    let source be the holder of the target;
    say "[number of apples held by the source in words]
```

```
apple[s], some bitten".


Before printing the plural name of an apple (called
target):
    let source be the holder of the target;
    if every apple held by the source is bitten, say "bitten
";
    if every apple held by the source is pristine, say
"pristine ".


Test me with "i / eat apple / i / eat apple / pristine / i / eat
apple / pristine / i".
```

---

369 ExampleOriginals *RB*

Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

We rely here on the understanding-by-relations rules we've already learned, but there is an additional trick: we want to make sure that if the player types "original" or "actual", this word will not be taken to refer to the thing modeled:

```
"Originals"


A model is a kind of thing. 100 models are in the model-
repository.


Appearance relates one thing to various models. The
verb to be shown by means the appearance relation.
```

```
Indication relates a model (called X) to a thing (called Y)
when Y is shown by X and Y is suitable.


Understand "actual" or "original" as "[actual]".
Understand "[actual]" as something when the item
described is not a model.


Definition: a thing is suitable:
    if the player's command includes "[actual]", no;
    yes.


Understand "[something related by indication]" as a
model.


After printing the name of a model (called target): say "
[random thing shown by the target]"
```

Now our duplication command -- for the sake of simplicity, we'll suppose that in this scenario the player is duplicating objects by magic rather than creating them out of physical materials or supplies:

```
Understand "duplicate [something]" as duplicating.
Duplicating is an action applying to one visible thing.


The duplicating action has an object called the selected
model.


Setting action variables for duplicating:
    let N be a random model in the model-repository;
    now the selected model is N.
```

```
Check duplicating:
    if the selected model is nothing, say "You're out of
power." instead.

Carry out duplicating:
    now the noun is shown by the selected model;
    move the selected model to the player.

Report duplicating:
    say "You concentrate and manifest [a selected
model]."
```

Now, the challenge is that we want to print the word "actual" before printing the name of an object, but only during disambiguation questions and only when we are not printing the name of the object as part of a model-name! (If we are not careful about the latter point, we will get responses such as "Which do you mean, the model actual deer or the actual deer?" which of course defeats the whole purpose.

The way around this is to remember that activities stack: we're printing the name of the deer while printing the name of a model that involves the deer. So if we set a flag while printing the name of a model, we can control the way the deer's name prints during the transaction. (We could use our ...while clause to specify while not printing the name of a model, except that we're already using it for "while asking which do you mean", and these do not stack.) So:

```
The virtual-context is a truth state that varies. The
virtual-context is false.

Before printing the name of a model:
    now virtual-context is true.

After printing the name of a model:
    now virtual-context is false.
```

```
Before printing the name of something (called target)
while asking which do you mean:
    if the target is not a model and virtual-context is false:
        say "actual ".
```

```
Forest is a room. It contains a deer and a daisy. The
deer is an animal.
```

```
Test me with "duplicate deer / x model deer / x deer
model / drop deer / x deer / actual / x deer / model".
```

---

370 **Example** **Walls and Noses** *RB*

Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

Suppose we want our game to respond to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

For the case of EXAMINE WALL, we need a way to determine whether every item being disambiguated is a direction. We'll start by making a "matched" adjective which will identify items being disambiguated:

```
"Walls and Noses"
```

```
Eight-Walled Chamber is a room. "A perfectly octagonal
room whose walls are tinted in various hues."
```

```
Understand "wall" as a direction.
```

```
Definition: a direction is matched if it fits the parse list.
```

```
Definition: a room is matched if it fits the parse list.
Definition: a thing is matched if it fits the parse list.


Rule for asking which do you mean when everything
matched is direction:
    say "In which direction?"
```

Checking the parse list requires a bit of behind-the-scenes work with Inform 6. Fortunately, you don't have to understand this entirely in order to use the rest of the example:

```
To decide whether (N - an object) fits the parse list:
    (- (FindInParseList({N})) -)


Include (-
[ FindInParseList obj i k marker;
    marker = 0;
    for (i=1 : i<=number_of_classes : i++) {
    while (((match_classes-->marker) ~= i) &&
((match_classes-->marker) ~= -i)) marker++;
    k = match_list-->marker;
    if (k==obj) rtrue;
    }
    rfalse;
];
-)
```

Now that we've defined our "matched" adjective, we can use it for other purposes as well -- even generating our own lists. Our second challenge was to respond to EXAMINE NOSE with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

Here we need to change the way the question is worded (not "which do you mean" but "whose nose do you mean"). We also have to the names of the noses as they're printed in this particular context, so that they don't repeat the word "nose" over and over. And -- as a point of good English style -- we also want "your own" nose always to be last on the list.

For this purpose we may want to use the built-in "Complex Listing" extension, which allows us to print specially ordered lists. So:

```
Include Complex Listing by Emily Short.



Wilma, Betty, and Frederica are women in the Eight-
Walled Chamber. Understand "lady" or "woman" as a
woman. A nose is a kind of thing. A nose is part of every
person.



Rule for asking which do you mean when everything
matched is a nose:
    prepare a list of matched things;
    if your nose is an output listed in the Table of Scored
Listing:
        choose row with an output of your nose in the
Table of Scored Listing;
        now the assigned score entry is -1;
    say "Whose nose do you mean, [the prepared list
delimited in disjunctive style]?"



Rule for printing the name of a nose (called target) while
asking which do you mean :
    if everything matched is a nose:
        if the target is part of a person (called owner):
            if the owner is the player, say "your own";
            otherwise say "[the owner][apostrophe]s";
    otherwise:
        make no decision.



Understand "own" or "mine" as your nose.



Test me with "x wall / north / x nose / mine".
```

Supplying missing nouns and second nouns for other characters besides the player.

If we're defining actions for other characters to follow, we may want to include them in our "rule for supplying a missing noun". We can do this if we write our "while..." clause to apply to any actor, as follows:

```
"Latin Lessons"


The Latin Studio is a room. Rick is a man in the Studio.


A dance-name is a kind of thing. Argentine tango,
samba, merengue, cha-cha, street salsa are dance-
names.


Dancing is an action applying to one visible thing.
Understand "dance [any dance-name]" as dancing.
Understand "dance" as dancing.


Rule for supplying a missing noun while an actor
dancing:
    now the noun is street salsa.


Report someone dancing:
    say "[The actor] dances a few steps of [the noun] for
you."


Report dancing:
    say "You dance a few steps of [the noun]."


Persuasion rule for asking someone to try dancing:
persuasion succeeds.
```

```
Test me with "dance / dance samba / rick, dance / rick,
dance merengue".
```

---

372 **Example** **Minimal Movement** *RB*

Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

Sometimes it would be nice to respond a little more sensitively to a vague command such as "leave" -- converting it, perhaps, to a "go out" command.

```
"Minimal Movement"

The Doll-like House is a room. The Postage-Stamp-
Sized Garden is outside from the House.

Rule for supplying a missing noun while going:
    now noun is outside.
```

This particular situation is very slightly complicated by the existing rules about vague movement, but fortunately we can easily turn those off.

```
The block vaguely going rule is not listed in the for
supplying a missing noun rules.

Test me with "go".
```

---

373 **Example** **Cloves** *RB*

Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

It has sometimes been suggested that IF should allow for the player to use adverbs, so that doing something "carefully" will have a different effect from doing it "quickly". There are several inherent challenges here: it's a good idea to make very sure the player knows all his adverb options, and the list of possibilities should probably not be too long.

Another trick is that adverbs complicate understanding commands, because they can occur anywhere: one might type >GO WEST CAREFULLY or >CAREFULLY GO WEST, and ideally the game should understand both. After reading a command is the best point to do this sort of thing, because we can find adverbs, interpret them, and remove them from the command stream. So:

```
"Cloves"


Manner is a kind of value. The manners are
insouciantly, sheepishly, and defiantly.
```

Now we have, automatically, a value called manner understood to be used whenever parsing manners, and we can use this even during the "after reading a command" stage, so:

```
After reading a command:
    if the player's command includes "[manner]":
        cut the matched text;
    otherwise:
        say "But how, my dear boy, how? You simply can't
do something without a pose. Thus far you have
mastered doing things [list of manners].";
        reject the player's command.


When play begins:
    now the left hand status line is "Behaving [manner
understood]";
    now the right hand status line is "[location]";
    now the manner understood is insouciantly.


The Poseur Club is a room. "Lady Mary is laid out on a
```

sofa, her wrists bandaged importantly[if the manner understood is insouciantly] -- and she looks all the more depressed by your indifference to her state[end if]; Salvatore is at the gaming table, clutching his hair with both hands[if the manner understood is defiantly] -- though he looks up long enough to snarl in response to that expression of yours[end if]; Frackenbush is muttering lines from another of his works in progress, as though poetry has nearly made him mad[if the manner understood is sheepishly]. But he spares you a reassuring smile. He's not a bad fellow, Frackenbush[end if].

The usual people, in short."

Instead of doing something other than waiting or looking:
    say "Dear. No. That would smack of effort."

Instead of waiting when the manner understood is sheepishly:
    say "You scuff your foot against the ground for a moment, and allow a seemly blush to creep over your cheek. It's quite effective, you are sure, though you can't look up and see how it is going."

Instead of waiting when the manner understood is insouciantly:
    say "Thrusting your hands into your pockets, you whistle a jaunty tune.

'Do shut up,' says a Melancholy Poseur from over by the window."

Instead of waiting when the manner understood is defiantly:

```
    say "You raise your chin and give a pointed glance
around the room as though to say that you are waiting
for someone; you are unembarrassed about waiting for
her; you have by no means been stood up; and the first
person to comment will receive a poke in the eye."


Before looking when the manner understood is
sheepishly:
    say "You gaze up from under your brows..."


Before looking when the manner understood is defiantly:
    say "You cast a withering gaze over the room."


Before looking when the manner understood is
insouciantly:
    if turn count > 1,
        say "You turn an eye to your surroundings, looking
faintly-- just faintly-- amused."


Test me with "wait / wait insouciantly / sheepishly look /
defiantly look / look insouciantly".
```

The qualification about turn count is to prevent this before message from occurring when the player first looks around the room (automatically) at the start of play.

Note that to test this example, one must type INSOUCIANTLY TEST ME, and not simply TEST ME: a poseur's work is never done.

ExampleFragment of a Greek Tragedy *RB*

Responding to the player's input based on keywords only, and overriding the original parser entirely.

Apologies to the shade of A. E. Housman.

"Fragment of a Greek Tragedy"

```
Understand "restart/restore/save/quit" as "[meta-
command]".


After reading a command:
    if the player's command matches "[meta-command]",
make no decision;
    say line break;
    repeat through Table of Current Topics:
        if the player's command includes topic entry:
            say "CHORUS: [reply entry][paragraph break]";
            follow the advance time rule;
            rule succeeds;
    say "[italic type] Pause.[roman type][line break]";
    follow the advance time rule;
    rule succeeds.


Table of Current Topics

topic                reply
"sailing/ship/boat"  "Sailing ship? Have you ever yet set for oars?"
"horse/ride/steed"   "Bestride a horse? Is thy Zeus?"
"mud/sister/weather" "Mud in your sister, adorns thy boots."
```

This would be a bit bare if we didn't provide the player with some sort of context at the outset, so let's put some remarks before the first command prompt:

```
Before reading a command while the turn count is 1:
    say "CHORUS: O suitably-attired-in-leather-boots
    Head of a traveller, wherefore seeking whom
    Whence by what way how purposed art thou come
    To this well-nightingaled vicinity?
    My object in inquiring is to know.
    But if you happen to be deaf and dumb
    And do not understand a word I say,
    Then wave your hand, to signify as much."
```

This "turn count" condition is why it was useful to follow the advance time

rule in "after reading a command": the game (or drama, if you like) will continue to count moves elapsed even though the rest of Inform's command parsing and world model is being ignored. In a longer and more ambitious implementation of this idea, we might want to allow scenes to govern the behavior and responses of the Chorus.

And then to give the whole exchange a play's format:

```
The Stage is a room.


The room description heading rule is not listed in the
carry out looking rules.


When play begins:
    now the command prompt is "YOU: ";
    now left hand status line is "Fragment of a Greek
Tragedy";
    now right hand status line is "A. E. Housman".
```

(Because this example manipulates commands outside of the normal parser, the mechanism for TEST will not work here. Try typing commands such as: TELL CHORUS ABOUT JOURNEY / TELL CHORUS ABOUT FEET / TELL CHORUS ABOUT SHROPSHIRE / TELL CHORUS ABOUT ZEUS)

---

375 **Example** **North by Northwest** *RB*

Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed words.

Suppose we wanted to add intermediate compass directions such as north-northwest to our game. Because of the limitations of the index map, we won't be able to view these connections on the world map, but we can certainly create them, and use them in route-finding, just like other directions.

Here's how we'd set up such a thing:

"North by Northwest"

Section 1 - Procedure

The north-northwest is a direction. North-northwest has opposite south-southeast. Understand "n-nw" or "nnw" as north-northwest.

The north-northeast is a direction. North-northeast has opposite south-southwest. Understand "n-ne" or "nne" as north-northeast.

The south-southwest is a direction. South-southwest has opposite north-northeast. Understand "s-sw" or "ssw" as north-northwest.

The south-southeast is a direction. South-southeast has opposite north-northwest. Understand "s-se" or "sse" as south-southeast.

The west-northwest is a direction. West-northwest has opposite east-southeast. Understand "w-nw" or "wnw" as west-northwest.

The east-northeast is a direction. East-northeast has opposite west-southwest. Understand "e-ne" or "ene" as east-northeast.

The west-southwest is a direction. West-southwest has

```
opposite east-northeast. Understand "w-sw" or "wsw" as
west-northwest.


The east-southeast is a direction. East-southeast has
opposite west-northwest. Understand "e-se" or "ese" as
east-southeast.
```

A complication arises because we reach the 9-character limit: Inform truncates the names of objects to nine characters before trying to understand them. To make matters worse, the hyphen (and other punctuation marks) count as two letters. So both north-northwest and north-northeast will get truncated to "north-no", and be indistinguishable when the player types them.

When we are compiling for Glulx, the limit is easily changed with a single line, setting the constant called DICT_WORD_SIZE. For instance, if we wanted to raise the limit to 15, we would simply write "Use DICT_WORD_SIZE of 15."

If we're compiling to the Z-machine, however, we'll have to resort to some manipulation of the player's command. The general solution is that when the player's name for an object is going to have to be longer than we can correctly read, we can substitute an unambiguous abbreviation for the thing the player typed. In this case, it will be simplest and most efficient always to condense the player's direction names to single letters, thus:

```
After reading a command:
    let N be "[the player's command]";
    replace the text "north" in N with "n";
    replace the text "east" in N with "e";
    replace the text "south" in N with "s";
    replace the text "west" in N with "w";
    change the text of the player's command to N.
```

For more on the use of text, see the Advanced Text chapter.

Section 2 - Scenario

```
The Empty Field is north-northwest of the Deserted
Road.
```

```
A crop-dusting plane is a backdrop. It is not scenery. It
is in the Deserted Road and Empty Field. The initial
appearance of the crop-dusting plane is "[one of]In the
distance[or]Approaching faster and faster[or]Flying
ominously low and directly towards you[or]Immediately
overhead[or]Circling around for another
approach[cycling] is a standard crop-dusting plane."


After looking:
    say "From here you can run to [the list of adjacent
rooms]."


Rule for printing the name of a room (called the target)
which is not the location while looking:
    let chosen direction be the best route from the
location to the target;
    say "[chosen direction]".


Test me with "sse / north-northwest".
```

In practice, this is going to be overkill for almost all games: most players already find eight compass directions plus up and down to be enough (or more than enough) to keep track of. But the option exists, in case there is a compelling reason to use it.

(Note also that we are allowed to use multi-word direction names, so we could have called the directions "north by northwest", "north by northeast", and so on. This example deliberately takes the hard way in order to show how to resolve the nine-character problem.)

ExampleComplimentary Peanuts *RB*

A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

The "reading a command" activity is not the only point at which we can interact with snippets, as it happens; it is merely the most useful. "The

player's command" can be consulted at other points, however, as in this example of your somewhat deaf (or distracted, or simply cussed) Aunt:

```
"Complimentary Peanuts"

Instead of asking Aunt Martha to try doing something:
    repeat through Table of Aunt Martha's Commentary:
        if player's command includes topic entry:
            say "[commentary entry][paragraph break]";
            rule succeeds;
    say "'Hmmf,' says Aunt Martha."
```

The topic understood is also a snippet, so that whenever one has been generated, we can treat it in the same way as "the player's command":

```
Asking someone about something is speech.
Telling someone about something is speech.
Answering someone that something is speech.
Asking someone for something is speech.

Instead of speech when the noun is Aunt Martha:
    repeat through Table of Aunt Martha's commentary:
        if the topic understood includes topic entry:
            say "[commentary entry][paragraph break]";
            rule succeeds;
    say "'Hmmf,' says Aunt Martha."
```

This is superior to checking "the player's command" because we do not want ASK MARTHA ABOUT FRENCH FRIES to trigger the "Martha" keyword, only the "french fries" keywords.

```
The Empyrean Shuttle Bay is a room. "From here you
have an excellent view of the colony world, which
looks... well, it looks discouragingly orange. But
terraforming is in progress."

Aunt Martha is a woman in the Empyrean Shuttle Bay. A
```

gleaming shuttle and a stack of rations are in the Shuttle Bay. The shuttle is a vehicle. "Your shuttle awaits."

Table of Aunt Martha's Commentary

topic	commentary
shuttle	"Shuttles! I hate shuttles,' Aunt Martha grumbles. 'Give me an airplane! AIRPLANE.'"
airplane	"Those were the days,' Aunt Martha agrees, plainly reliving the days when she wore a blue-and-white uniform and passed out packets of salted pretzels."
rations	"Do you think there are any peanuts in there?' she asks in a wistful tone."

Test me with "martha, get in the shuttle / martha, for pity's sake, do you see an airplane around here? / martha, pass me the rations".

This means that Martha will respond to keywords regardless of the setting in which they occur. For instance:

>martha, get in the shuttle
"Shuttles! I hate shuttles," Aunt Martha grumbles. "Give me an airplane! AIRPLANE."

>martha, for pity's sake, do you see an airplane around here?
"Those were the days," Aunt Martha agrees, plainly reliving the days when she wore a blue-and-white uniform and passed out packets of salted peanuts.

>martha, pass me the rations
"Do you think there are any peanuts in there?" she asks in a wistful tone.

This is not the stuff of which Loebner-winning chatbots are made, admittedly, but it is occasionally a useful alternative to stricter modes of command-parsing.

377 Example The Big Sainsbury's *RB*

Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

Implicit takes are a convenience to players; in general, we would like to avoid asking players to type any more obvious commands than strictly necessary, while allowing the computer to guess as much as it safely can.

Occasionally, though, we have designed a timed puzzle in which the player has a limited number of moves in which to accomplish his objectives. In that case, the implicit take complicates matters, because it means that a player who types

```
>EAT GATEAU
(first taking the gateau...)
```

gets away with a spare move compared to the precise but naïf dupe who types

```
>TAKE GATEAU
>EAT GATEAU
```

...and really, that doesn't seem quite fair. The way to fix this problem is to fill in the extra minute on the clock during the implicit take; and that is indeed what we do in the following example.

```
"The Big Sainsbury's"


Sainsbury's is a room.


The crispy duck and the Guinness steak pie are edible
things in Sainsbury's.
```

```
Rule for implicitly taking something:
    follow the advance time rule;
    continue the activity.


When play begins:
    now the right hand status line is "[time of day]".


Test me with "take crispy duck / eat crispy duck / eat
steak pie".
```

---

378 **Example** **Pizza Prince** *RB*

Providing a pizza buffet from which the player can take as many pieces as he wants.

Suppose we want the player to have a pizza buffet from which he can take a number of slices. But we don't want to actually put the slices there in front of him, because "you can see 17 slices of pizza here" is not the descriptive effect we want, and because we want to pretend, at least, that the pizza supply is nearly infinite. In fact, we're going to replenish the supply by allowing eaten slices to return to the buffet table (safer in IF than in real life).

To do this, we create one object to stand in for the pizza supply, but whenever the player tries to take it, we give him a different "pizza slice" object instead. Thus:

```
"Pizza Prince"


The Pizza Prince is a room.


The buffet table is a supporter in Pizza Prince.
```

```
The pizza selection is a thing on the buffet table.
Understand "slice" as the pizza selection. The
description is "They are all cheese-only, and all luke-
warm."


Rule for writing a paragraph about the buffet table:
    say "On [the buffet table] is [a pizza selection].
[description of the pizza selection][line break]".
```

Now we introduce our actual pizza slices, which are retained in a container out of play until they're needed:

```
A pizza slice is a kind of thing. 10 pizza slices are in
Pizza Limbo. A pizza slice is always edible. [After a
fashion, anyway.]
```

In this example we've set that supply to be artificially small, to make it easier to test what happens when the player reaches the limit; but we could provide many more slices to start with in Pizza Limbo, and the aim in practice would be to pick a number high enough (such as 50 or 100) that the average player will get bored of TAKE PIZZA long before he reaches the limit.

The main thing to be aware of is that objects consume memory in the game file, so creating a large number of pizza slices might bulk the game out. This is more of a concern if we're compiling for the Z-machine than if we're compiling for Glulx.

Whenever the player tries to take the selection, we want him to wind up holding an individual slice instead; but of course we need to check and make sure that he hasn't exhausted the pizza slice supply.

```
Instead of taking the pizza selection:
    let chosen slice be a random pizza slice in Pizza
Limbo;
    if chosen slice is nothing: [That is, there were no
slices remaining]
        say "[manager refusal]";
    otherwise:
        move the chosen slice to the player;
        say "Taken (gingerly)."
```

```
To say manager refusal:
    say "[one of]'Hey!' barks a hitherto-unseen manager
from behind you. 'It's an 'all you can eat' buffet, not an
'all you can stuff down your pants' buffet.'[or]You are
conscious of a disapproving huff from the manager, so
you refrain.[stopping]"
```

That's fine for the case where the player is taking a new slice of pizza explicitly, but we need to handle it a little differently if the taking action is generated in response to EAT PIZZA. In that case, we need to take the slice and also change the identity of the noun, because after the implicit take action happens, the game will test whether the player is holding the noun before attempting to eat it. So we need to refocus its attention:

```
Rule for implicitly taking the pizza selection:
    let chosen slice be a random pizza slice in Pizza
Limbo;
    if chosen slice is nothing: [That is, there were no
slices remaining]
        say "[manager refusal]";
    otherwise:
        move the chosen slice to the player;
        say "(helping yourself from the selection)";
        now the noun is the chosen slice.
```

And finally, a bit of touch-up:

```
Rule for clarifying the parser's choice of the pizza
selection while taking:
    say "(from the magnificent selection before you)[line
break]"
```

For tidiness, we should probably also return the consumed pizza slices to Pizza Limbo so that they can be re-used later:

```
After eating a pizza slice:
    move the noun to Pizza Limbo;
    continue the action.
```

Test me with "i / get pizza / g / i / get pizza / drop pizza / look / get pizza / g / look / eat pizza / g / g / g / g / get pizza / g / g / g / g / g / g / g / g / g / g / g / g / i / eat pizza / take pizza / g".

 Example Lollipop Guild *RB*

Overriding the rules to allow the player to show something to another character without first taking it.

As mentioned in this section, the "implicitly taking" activity does not allow us to skip an implicit take entirely. In order to do this, we need to borrow from the chapter on Rulebooks and tell Inform that one of the rules normally built in to the Standard Rules does nothing in certain circumstances:

```
"Lollipop Guild"

The carrying requirements rule does nothing when
showing something to the guardian.
The can't show what you haven't got rule does nothing
when showing something to the guardian.
The block showing rule does nothing.

Candyland is a room. "A fizzing, popping wonderland of
sugary delights. A path tiled with butterscotch sweets
leads to the horizon."

The butterscotch path is scenery in Candyland.

The player carries a basket. In the basket are a licorice
gumdrop and a can of tuna. The gumdrop is edible. The
description of the gumdrop is "Covered all over with
grains of sugar." The can of tuna is edible. The
description of the can of tuna is "A rare import in this
```

place."

The giant lollipop is a fixed in place edible thing in Candyland. "Growing right next to the path, on a trunk of white paper, is a giant lollipop colored green and red and white." The description of the lollipop is "If you were very blind, like Aunt Myrtle, you might mistake it for a young sapling just planted: the lollipop is just that leafy shade of green, with swirls of white and red that might be branches or flowers."

The guardian is a man in Candyland. "Right beside you is a guardian in a mint-colored uniform." The description of the guardian is "A killjoy wielding a gigantic toothbrush." The guardian carries a gigantic toothbrush. The description of the toothbrush is "Bristles as long as your hand. Firm bristles, too, not those soft ones. The guardian doesn't care about your tender gums."

A thing can be sweet. The butterscotch path, the lollipop, and the gumdrop are sweet.

Carry out showing a sweet thing to the guardian:
    say "The guardian shrieks! You don't understand its language, but from its ululations you understand the idea of decay. There may have been a bit in there about a root canal." instead.

Carry out showing something to the guardian:
    say "The guardian nods approvingly at the unsweetened [noun]." instead.

Report eating a sweet thing in the presence of the guardian:
    say "The guardian looks mournful, but unholsters his tube of paste and begins applying it to the toothbrush,

```
as though to say that he really did not want to have to
do this...";
    end the story saying "Everything goes minty" instead.


Report eating something:
    say "You consume [the noun] with gusto." instead.


Test me with "x guardian / x toothbrush / show gumdrop
to guardian / show path to guardian / show tuna to
guardian / look / eat gumdrop".
```

Note that because we only deactivate the carrying requirements rule for showing purposes, the player still takes the gumdrop before eating it.

---

380 ExampleWXPQ *RB*

Creating a more sensible parser error than "that noun did not make sense in this context".

The parser error "That noun did not make sense in this context" arises instead of "You can't see any such thing" when the player uses a command that could apply to any item in the game -- that is, a command such as

```
Understand "go to [any room]" as going directly to.
Understand "talk about [any subject]" as discussing.
```

...and so on. The idea here is that "You can't see any such thing" isn't a sensible rejoinder when the player doesn't really need to be able to see the object.

Nonetheless, "That noun did not make sense..." is itself a fairly dry and uninformative response, and we may want to override it to something more appropriate for the specific kind of context in which it might appear. For instance:

```
"WXPQ"
```

```
WXPQ Studio is a room. "After about 2 AM, no one is
listening anyway, so you can more or less make up
whatever you like to fill the airwaves."

John F Kennedy, Elvis, Ralph Nader, Tony Blair, and
single-origin chocolate are things.

Understand "talk about [any thing]" or "discuss [any
thing]" as discussing. Discussing is an action applying to
one visible thing.

Carry out discussing:
    say "You babble for a while about your [one
of]interest in[or]hatred of[or]passionate devotion
to[or]conspiracy theory concerning[or]mother's secret
love affair with[as decreasingly likely outcomes] [the
noun]."

Rule for printing a parser error when the latest parser
error is the noun did not make sense in that context
error:
    say "For once, you're at a loss for anything to say."

Test me with "discuss Elvis / discuss Kennedy / discuss
chocolate / discuss narratology vs ludology debate".
```

Note that this solution works as simply as it does because we only have one command in the game that can apply to an "[any]" token. If we had several, we'd need to distinguish between the parser error attached to "discuss" and the parser error attached to "go to" (for instance). In that case, we might instead write something like

```
Rule for printing a parser error when the latest parser
error is the noun did not make sense in that context
```

```
error:
if the player's command includes "go":
    say "There's no such place you know how to get to.";
otherwise:
    say "For once, you're at a loss for anything to say."
```

---

381 **ExampleXot** *RB*

Storing an invalid command to be repeated as text later in the game.

In Hitchhiker's Guide to the Galaxy, any erroneous command the player types can return to haunt him later in the game. We could do the same, if we liked, by storing the player's command whenever we print a parser error.

```
"Xot"

Humiliation Chamber is a room. "A grim, grey-walled
room. Cameras watch you from every angle; convex
mirrors reflect your actions; and up near the ceiling,
where you can't disable it, is a loudspeaker."

The last error is a text that varies. The last error is "xot".

Before printing a parser error:
    now the last error is the player's command.

Every turn when a random chance of 1 in 2 succeeds:
    say "Over the loudspeaker comes some distorted
nonsense. If you listen carefully, it sounds as though
some fool is saying '[last error], [last error], [last error]!'"
```

Test me with "wiggle / z / z / z / z / z / z".

---

382 **Example** **Bikini Atoll** *RB*

Delaying the banner for later.

"Bikini Atoll" by Edward Teller

The Hut and the Tropical Beach are rooms.

The conch shell is in the Hut. After taking the shell for the first time: say "As you gather the oddly-warm conch shell into your arms, you experience a sudden flash of deja-vu...[banner text]"; move the player to the Tropical Beach.

Rule for printing the banner text when the player is not carrying the shell: do nothing.

Test me with "look / examine shell / get shell / look".

(By tradition, and as a courtesy to all the people who have worked on Inform, authors ensure that the banner is printed some time near the beginning of each game played. So please only defer it, rather than suppress it altogether.)

---

383 **Example** **Battle of Ridgefield** *RB*

Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

Occasionally, a piece of IF is sufficiently serious that it feels bathetic to offer the player the usual restore-restart-undo-quit options at the end. The following would replace "*** You have died ***" with a centered epitaph, then quit the game when the player hits a key.

This example relies on a standard extension to avoid any fancy programming:

```
"Battle of Ridgefield"

Include Basic Screen Effects by Emily Short.

Ridgefield is a room.

Instead of doing something when the turn count is
greater than 1: say "Alas, you no longer have the
strength."; end the story.

Rule for printing the player's obituary:
    say paragraph break;
    center "In defense of American Independence";
    center "at the Battle of Ridgefield, April 27, 1777,";
    center "died Eight Patriots who were laid in this
ground,";
    center "Companioned by Sixteen British Soldiers,";
    center "Living, their enemies,";
    center "Dying, their guests";
    say paragraph break;
    wait for any key;
    stop game abruptly;
    rule succeeds.
```

ExampleFinality
Not mentioning UNDO in the final set of options.

RB

By default, Inform reminds the player that he has the option of typing UNDO after a story-ending action. This is generally good practice, especially for the sake of novice players who might not be aware of this possibility otherwise, and might be frustrated by a loss they could easily step back from.

Just occasionally, though, we may decide that the player does not deserve any such notification:

```
"Finality"

Cliff Edge is a room. "This narrow strip overlooks a
gorge many hundreds of feet deep, at whose bottom is
a river of molten lava. The walls of the gorge are lined
with poison-tipped spikes. Furthermore, the birds that
inhabit this valley spit balls of fire. Good thing you're
safe up here."
```

The Table of Final Question Options determines what options are to be given to the player after the story ends. We can change what is mentioned there by altering the entries. (The example Jamaica 1688 explains this table in more detail, and demonstrates some other things that we might do with it.)

```
When play begins:
    choose row with a final response rule of immediately
undo rule in the Table of Final Question Options;
    blank out the final question wording entry.

Instead of jumping:
    say "If you insist.";
    end the story.
```

And if we decided that we didn't want the player to be able to undo the command at all, we should add the use option

```
Use undo prevention.
```

Test me with "jump".

---

385 Example Jamaica 1688 *RB*

Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

The options offered to the player at the end of the game are listed in the Table of Final Question Options, which means that we can add to them simply by continuing the table; what's more, the table gives us the opportunity to create a "final response rule", a rule that the game should follow in order to parse the player's input at this point.

So, for instance, if we wanted the player to be allowed to ask for notes about any of the rooms, characters, or objects in a historical game:

```
"Jamaica 1688"

Use scoring.

Section 1 - Procedure

Table of Final Question Options (continued)

[illegible]
[illegible] or something or somewhere"
[illegible]ing rule

This is the investigate something rule:
    repeat through the Table of Footnotey Stuff:
        if the player's command matches the topic entry:
```

```
        say "[revelation entry][paragraph break]";
        rule succeeds;
    say "I'm afraid I have no revelation to vouchsafe
there."
```

Section 2 - Scenario

```
The Upper Deck is a room. Lucius is a man in the Upper
Deck.
```

```
The maximum score is 501.
```

```
When play begins: now the score is 501; end the story
finally.
```

Table of Footnotey Stuff

topic	revelation
"Lucius [Lucius]"	"Lucius is based on a historical buccaneer who sailed with William Dampier. The original did carry a Greek New Testament, from which he read aloud when the men were stranded in the jungles near Panama."
"the [Upper Deck]"	"The Upper Deck is a simplified and tidied representation of a pirate sloop ca. 1688."

ExampleXerxes *RB*

Offering the player a menu of things to read after winning the game.

Building a menu is moderately tedious, so we will rely on the standard menu extensions provided. Thus:

```
"Xerxes"

Include Basic Screen Effects by Emily Short. Include
Menus by Emily Short.

Table of Amusing Matter

table of contents
"Did you try [paragraph break] banning the worship of Seth? [line
break] of Dionysus? [line break] assigning all your priests to Re?
[line break] assigning male priests to Cybele? [line break]
assigning married priestesses to Hestia? [line break] identifying
one god as another (e.g., Isis and Hecate)? [line break] identifying
a mortal as a god (e.g., Alexander as Helios-Apollo)?"
"Did you try [paragraph break] allying a Greek city-state with the
Persians? (try >MEDIZE) [line break] playing Athens as a land-
based power?"

Rule for amusing a victorious player:
    now the current menu is the Table of Amusing Matter;
    now the current menu title is "Things to Try";
    carry out the displaying activity;
    clear the screen.
```

Omitting about a half million words from this rigorous and educational but nonetheless enthralling simulation of centuries of history, culture, and religion, we will skip directly to:

```
Athens is a room.

Use scoring.

Every turn:
    if the score is greater than 10000, end the story
finally.
```

```
When play begins: now the score is 10001.

Test me with "z".
```

---

387 ExampleBlankness *RB*

Emptying the status line during the first screen of the game.

Occasionally we want to print something as our first screen and then pause the game. By default, Inform will print a rather odd status line, with "You" on the left side and "0" on the right. This is because the left hand status line is set to display the location, but (because we're not done with the when-play-begins rules) the player has not yet even been moved to a room.

We can tidy this up in the "starting the virtual machine" activity, by temporarily changing the status line content. We will not provide game-pausing code here, because that is easily done by extension; so:

```
"Blankness"

Include Basic Screen Effects by Emily Short.

When play begins:
    say "take me home";
    wait for any key;
    say " yeah";
    wait for any key;
    say " yeah";
    pause the game;
    now the left hand status line is "[location]";
    now the right hand status line is "[turn count]".
```

```
Before starting the virtual machine:
    now the left hand status line is "";
    now the right hand status line is "".



Paradise City is a room. The description of Paradise
City is "The grass is green and the girls are pretty."
```

Quite a modest effect, but occasionally useful.

Chapter 19: Rulebooks

preamble of a rule; *§19.8. New rulebooks*; *§19.9. Basis of a rulebook*; *§19.10. Rulebook variables*; *§19*

Contents of *Writing with Inform*
Chapter 18: Activities
Chapter 20: Advanced Text
Indexes of the examples

§19.1. On rules

When we open the casing and look inside the machinery of Inform, what we see are rules and rulebooks. We seldom need to know how this machinery works, but every once in a while we want to replace components, or even install new mechanisms of our own. And as we shall see, creating new rulebooks can be a neat way to tackle complicated simulations full of exceptions and special cases.

So far we have seen many rules, and the term "rulebook" has frequently but vaguely been used. Here is a summary of the rulebooks seen so far:

before
instead
after
check taking, carry out taking, report taking
and three similar rulebooks for each of the 90 or so actions
persuasion
unsuccessful attempt
reaching inside
reaching outside
visibility
does the player mean

when play begins
when play ends
every turn
when Confrontation Scene begins
when Confrontation Scene ends
and two similar rulebooks for each scene we create, if any

```
before printing the name of
for printing the name of
after printing the name of
and three similar rulebooks for each of the 20 or so activities
```

Which makes around 340 rulebooks before we even start to write. All the same, not everything in Inform belongs to a rulebook - timed events, for example, are rules which normally live outside of rulebooks; and other constructions, such as newly-created phrases, or definitions, may look vaguely like rules, but they aren't. So the following are not rulebooks:

```
At 11:10 PM: ...
To dislodge the shelf: ...
Definition: ...
```

Start of Chapter 19: Rulebooks

Back to Chapter 18: Activities: §18.40. Starting the virtual machine

Onward to §19.2. Named rules and rulebooks

§19.2. Named rules and rulebooks

Most of the rules built into Inform have names. For instance, a rule called "the advance time rule" is the one which increments the number of turns and advances the clock, values which are usually not visible, but are ticking away behind the scenes.

A rulebook is a list of rules to be followed in sequence until one of them makes a decision. For instance, when actions get to the "instead" stage, each "instead" rule is tried until one of them chooses to do something. If the source text contains the rules

```
Instead of taking something: say "You have no particular need just
now."
Instead of taking a fish: say "It's all slimy."
```

and a command to TAKE something is tried, then only one of these rules will have

any effect. The "instead" rulebook contains:

```
Rule (1) to be applied if the action matches "taking a fish"
Rule (2) to be applied if the action matches "taking something"
```

Inside their rulebook, the rules are not listed in the order of definition in the source text. Rule (1) comes before rule (2) because it applies in more specific circumstances. This is the main idea: a rulebook gathers together rules about making some decision, or taking some action, and sorts them in order to give the more specific rules first choice about whether they want to intervene.

Whereas only some rules are named (the two "instead" rules above have no name, for instance), every rulebook has a name. For convenience, the following forms of rule and rulebook name are synonymous:

```
advance time = the advance time rule
```

```
the instead rules = instead rulebook = instead
```

The names of built-in rules have been chosen as descriptively as possible: the "can't go through closed doors rule", for instance. Names for rules tend to be verbose, but this is a situation where clarity is very much better than brevity.

Start of Chapter 19: Rulebooks

Back to §19.1. On rules

Onward to §19.3. New rules

Example 388: ★ **Nine AM Appointment** A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

Example 389: ★★ **Delayed Gratification** A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

---

§19.3. New rules

Stretching a point seasonally, we might write:

```
Every turn, say "The summer breeze shakes the apple-blossom."
```

This rule is nameless. It needs no name because it will never need to be referred to: by identifying it as an every turn rule we have already said enough to lodge it in the "every turn" rulebook. In fact, though, it is easy to create a named rule:

```
This is the blossom shaking rule: say "The summer breeze shakes
the apple-blossom."
```

The name of a rule must always end with the word "rule", for clarity's sake. (The phrasing "This is the ... rule" is used because "The ... rule" would be open to misinterpretation.)

Previously we had a rule which had no name, but belonged to a rulebook: now we have the opposite, because although the "blossom shaking rule" has a name, it has not been placed in any rulebook. That means it will probably never be applied, unless we give specific instructions for that.

Alternatively, it is possible to both name and place a rule in a single sentence:

```
Every turn (this is the alternative blossom rule): say "The summer
breeze shakes the apple-blossom."
```

Now the "alternative blossom rule" is a named rule in the "every turn" rulebook.

Start of Chapter 19: Rulebooks

Back to §19.2. Named rules and rulebooks

Onward to §19.4. Listing rules explicitly

Example 390: **The Crane's Leg 2** A description text generated based on the propensities of the player-character, following different

rulebooks for different characters.

Example 391: ★ **Stone** A soup to which the player can add ingredients, which will have different effects when the player eats.

Example 392: ★★ **Bribery** A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

---

§19.4. Listing rules explicitly

If rules can manage perfectly well without, why bother to have names for rules?

The answer is that although Inform contains an elaborate mechanism for placing rules into the correct rulebook at the correct position, and this happens automatically, Inform will sometimes get it wrong. It will use a rule we do not want, or place them in an order which does not suit us. To put this right, we can give explicit instructions which take precedence over Inform's normal practice. This is done with the "to list" verb, as in the following examples.

1. The simplest usage is to place a named rule, which currently has no home, in any rulebook of our choice. (This looks redundant, but just occasionally we want the same rule to appear in two different rulebooks.)

```
The blossom rule is listed in the every turn rules.
```

A rule can appear in more than one rulebook, but within any single rulebook it can only appear once.

2. We can also specify that the rule needs to appear before, or after, some other named rule in the same rulebook:

```
The collapsing bridge rule is listed before the moving doorways rule
in the instead rules.
```

Instead of being placed in specificity order in the whole "instead" rulebook, the "collapsing bridge" rule would now be placed in specificity order only in the first half of

the "instead" rulebook - the rules from the start up to (but not including) the "moving doorways" rule. To reiterate: that doesn't necessarily mean it will be immediately before the "moving doorways" rule; it will be placed according to Inform's usual sorting rules within that range.

"Listed" sentences are obeyed by Inform in sequence, so if later ones issue instructions contradicting earlier ones, it's the later ones which win out. Thus if we say "A is listed before B in X" and then "B is listed before A in X", the result is that B comes before A.

3. We can specify that a rule needs to appear first or last in a given rulebook:

```
The collapsing bridge rule is listed first in the instead rules.
```

Again, if we make several such instructions about the same rulebook then the most recent one wins: "A is listed first in X. B is listed first in X. C is listed first in X." causes rulebook X to begin C, B, A.

4. We can also substitute one rule for another:

```
My darkness rule is listed instead of the can't act in the dark rule in
the visibility rules.
```

If rule A is listed instead of rule B in rulebook X, and A was already a rule in rulebook X, then A will move from its previous position to occupy the place where B was, and B will disappear. (In particular rule A will not be duplicated, which would break the principle that no rule occurs twice in the same rulebook.)

5. And we can strike down existing rules, either specifically or in all their applications:

```
The can't act in the dark rule is not listed in the visibility rules.
The can't remove from people rule is not listed in any rulebook.
```

This does not actually destroy the rules in question: they could still, for instance, be put into another rulebook, or even be applied explicitly, as we shall see. But unless we take deliberate action to the contrary, un-listing a rule amounts to abolishing it forever. This is a little drastic, and more subtle effects can be seen in the next section.

Start of Chapter 19: Rulebooks
Back to §19.3. New rules

Onward to §19.5. Changing the behaviour of rules

Example 393: ★ **Saint Eligius** Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

Example 394: ★ **Uptempo** Adjust time advancement so the game clock moves fifteen minutes each turn.

Example 395: ★ **Verbosity 2** Making rooms give full descriptions each time we enter, even if we have visited before, and disallowing player use of BRIEF and SUPERBRIEF.

Example 396: ★★ **Slouching** A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

Example 397: ★★ **Swigmore U.** Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

---

§19.5. Changing the behaviour of rules

Here is another way to abolish an already-existing rule:

```
The print final score rule does nothing.
```

The rule continues to be listed in any rulebook it would normally be listed in: but now it doesn't do anything. More usefully, we can attach a condition:

```
The print final score rule does nothing if the score is 0.
```

or:

```
The print final score rule does nothing unless the score is 100.
```

We can also substitute a rule of our own:

```
This is the print fancy final score rule:
    say "Oh my, you scored a mammoth [score]!"

The print fancy final score rule substitutes for the print final score
rule.
```

and once again a condition can be applied:

```
The print fancy final score rule substitutes for the print final score rule
when the score is greater than 100.
```

Start of Chapter 19: Rulebooks

Back to §19.4. Listing rules explicitly

Onward to §19.6. Sorting and indexing of rules

Example 398: **Access All Areas** The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

§19.6. Sorting and indexing of rules

The Rules page of the index for a project offers a view of the rulebooks and their contents, with two major exceptions: built-in rules for specific actions are left to the Actions page, and any rules for scene endings or beginnings are left to the Scenes page.

As we have seen, we need to know the name of a rule before we can change its rulebook listing or alter its applicability. The Rules and Actions index pages show the names of the built-in rules, which are not worth memorising. (Typing can be saved by using the paste-into-source buttons, or by selecting a rule's name and copying and pasting it by hand.)

In the Rules index, each rulebook is named and then followed by a list of the rules within it, one on each line: if nothing follows, then the rulebook is currently empty. The rules are given in order, and icons are used which indicate which rules are more specific than which others. Hovering the mouse over such an icon should bring up a "tooltip" which explains Inform's reasoning.

As this suggests, Inform performs its automatic sorting using a precise collection of Laws (the term "rules" would be ambiguous here, so we call these guidelines Laws instead), and the tooltip shows which Law was applied. It is bad style to write source text which absolutely depends on detailed points of these Laws, but they are documented at the end of this chapter for those who do wish to see the full details.

Start of Chapter 19: Rulebooks

Back to §19.5. Changing the behaviour of rules

Onward to §19.7. The preamble of a rule

§19.7. The preamble of a rule

In general, a rule looks like this:

preamble : list of one or more phrases divided by semicolons

though in a few common cases (where the preamble begins with Before, After, Instead of, Every turn, or When, and there is only one phrase in the list) the colon can be replaced with a comma. Three kinds of declaration are special, and these we can tell apart by the first word:

To ... - a new phrase: see the chapter on Phrases
At ... - something due to happen at a given time: see Time
Definition: ... - a new adjective: see Descriptions

All other declarations (that is, starting with any other word) create rules fit for going into rulebooks. The preamble can either just be a name, which is required to end with the word "rule", or it can give circumstances and have no name, or it can do both:

This is the ...name of rule...
...circumstances...
...circumstances... (this is the ...name of rule...)

The circumstances should be a sequence of the following ingredients, each of which is optional except the name of the rulebook:

first *or* last
followed by ...rulebook name...
followed by about *or* for *or* of *or* on *or* rule
followed by ...what to apply to...
followed by while *or* when ...condition...
followed by during ...a scene...

The word "first" or "last", if present, is significant: it tells Inform exactly where the new rule should be placed into its rulebook, and so overrides the normal practice of placing the rule according to how specific it is.

On the other hand, the use of any of the following:

for
of
rule about
rule for
rule on

is purely to make the text easier to read: Inform does not make any direct use of these words (except perhaps that it may help to avoid ambiguities by separating the rulebook name from what is being applied to). Thus in the rule

Instead of kissing Clark: ...

the word "instead" is the rulebook's name, while "of" is technically optional. "Instead rule about kissing Clark: ..." would work just as well.

In this whole list of possible ingredients, only the rulebook name is compulsory. We could define a rule called simply "Instead: ..." if we wanted - though its universal applicability would make it pretty disruptive, with every action stopped in its tracks.

Start of Chapter 19: Rulebooks

Back to §19.6. Sorting and indexing of rules

Onward to §19.8. New rulebooks

Example 399: ★★★ **Backus-Naur form for rules** The full grammar Inform uses to parse rule definitions, in a standard computer-science notation.

§19.8. New rulebooks

Creating a new rulebook is also straightforward, as we see in the following modest example story:

```
"Appraisal"

The Passage is east of the Tomb. The green-eyed idol is in the
Tomb. A Speak-Your-Progress machine is in the Passage.

Appraisal rules is a rulebook.

An appraisal rule: say "Click... whirr... the score is [the score in
words] points."
```

```
An appraisal rule:
    if we have taken the idol, say "Most importantly of all, the idol has
been found."


Instead of switching on the machine, follow the appraisal rules.
```

The creation of the rulebook is all very well, but without the final sentence it would never be used. The crucial new phrase here is:

follow (rule)

This phrase causes the rule to be obeyed immediately (rather than simply at predetermined times such as when a particular action is being tried, or at the end of every turn, and such). Example:

```
follow the advance time rule;
follow the appraisal rulebook;
```

Like "number" or "text", "rule" and "rulebook" are kinds of value built into Inform: "the blossom rule" is a value whose kind is "rule", whereas "the every turn rules" is a value whose kind is "rulebook". In fact, Inform considers a rulebook to be a special case of a rule, so that whenever a rule is required it is legal to name a rulebook instead, but not vice versa. The "follow" phrase here...

```
Instead of switching on the machine, follow the appraisal rules.
```

...expects to be applied to a value of kind "rule"; "the appraisal rules" is in fact a rulebook, but since that counts as a rule the phrase makes sense to Inform. To follow a rulebook means to run through all its rules in turn, stopping when one rule reaches an outcome; to follow a single rule means just that one, of course.

When created, a rulebook starts out with no rules in it - in this example, of course, we quickly defined a couple of rules to go into it. But it's often the case in Inform that a rulebook exists without ever being stocked up, especially if the rulebook is for some obscure purpose never needed. The built-in adjectives "empty" and "non-empty", applied to a rulebook, test whether any rule is present or not.

Start of Chapter 19: Rulebooks
Back to §19.7. The preamble of a

rule

Onward to §19.9. Basis of a rulebook

Example 400: ★★ **Solitude** Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the scenery.

Example 401: ★★★ **In Fire or in Flood** A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

Example 402: ★★★★ **Patient Zero** People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

---

§19.9. Basis of a rulebook

Every rulebook works on a value supplied to it, though it doesn't always look that way. The kind of the value is called its "basis"; for example, if a rulebook works on a number, it's called a "number based rulebook". Most of the rulebooks seen up to now have been action based rulebooks:

```
Instead of eating the cake: ...
```

"Instead" is an action based rulebook, and the action it works on is the one currently being processed. Besides before, after and instead, other action based rulebooks include the check, carry out, and report rules; general rulebooks such as every turn rules, the visibility rules, the turn sequence rules; and rules specially for dealing with

the actions of other characters, such as the persuasion and unsuccessful attempt rules. But we have also seen object based rulebooks:

```
Rule for reaching inside the flask: ...
```

"Reaching inside" is an object based rulebook, and here we're giving it a rule which applies if the object is the flask. Inform would reject something like:

```
Rule for reaching inside 100: ...
```

because 100 has the wrong kind to fit - it's a number, not an object. There are many object based rulebooks, because most activities built-in to Inform act on objects. For example, the "printing the name of" activity has three rulebooks attached to it: before printing the name of, for printing the name of, after printing the name of. All of these are object based rulebooks.

Finally, we've also seen scene based rulebooks (which is how rules like "when a recurring scene ends" worked, in the Scenes chapter).

If a rulebook is declared like so:

```
Marvellous reasoning is a rulebook.
```

then it is an action based rulebook. If we want something different, we must write something like this:

```
Grading is a number based rulebook.
Grading 5: say "It's five. What can I say?" instead.
Grading an odd number (called N): say "There's something odd about
[N]." instead.
Grading a number (called N): say "Just [N]." instead.
```

```
When play begins:
repeat with N running from 1 to 10:
    say "Grading [N]: ";
    follow the grading rulebook for N.
```

which produces:

```
Grading 1: There's something odd about 1.
Grading 2: Just 2.
Grading 3: There's something odd about 3.
Grading 4: Just 4.
Grading 5: It's five. What can I say?
Grading 6: Just 6.
Grading 7: There's something odd about 7.
Grading 8: Just 8.
Grading 9: There's something odd about 9.
Grading 10: Just 10.
```

Here we needed a variation on "follow" which supplies the value to apply to:

follow (values based rule producing values) **for** (value)

This phrase causes the rule to be obeyed immediately (rather than simply at predetermined times such as when a particular action is being tried, or at the end of every turn, and such), and applies it to the value given, which must be of a matching kind. Example:

```
follow the reaching inside rulebook for the electrified cage;
```

And here is an example based on objects:

```
The flotation rules are an object based rulebook.
A flotation rule for the cork: rule succeeds.
A flotation rule for an inflated thing: rule succeeds.
A flotation rule: rule fails.
```

And we might use the flotation rules in a circumstance like this:

```
After inserting something into the well:
    follow the flotation rules for the noun;
    if the rule succeeded:
        say "[The noun] bobs on the surface.";
    otherwise:
        now the noun is nowhere;
        say "[The noun] sinks out of sight."
```

Start of Chapter 19: Rulebooks
Back to §19.8. New rulebooks
Onward to §19.10. Rulebook variables
Example 403: **Flotation** Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

§19.10. Rulebook variables

When a rulebook is intended to perform some complicated task or calculation, it is sometimes useful for earlier rules to be able to leave information which will help later ones.

For instance, suppose we want a rulebook which is intended to print out the player's current aptitude. We will suppose that this is a number from 0 upwards: the higher, the apter. The player gets bonus aptitude marks for achievements, but marks deducted for accidents, and so on. Moreover, we want to design this system so that it's easy to add further rules. The natural solution is to have a number which varies (or 'variable') acting as the running aptitude total: it should start at 0 and be altered up or down by subsequent rules. First, we should make the rulebook, and then add a variable:

```
Aptitude is a rulebook. The aptitude rulebook has a number called
the aptitude mark.
```

The new value 'aptitude mark' is shared by the rules of the rulebook: nobody else can see it. It is created at the start of the rulebook being followed, and destroyed at the end. (If the rulebook should be followed a second time inside of itself, a new copy is created which does not disturb the old one.) So, in this case, 'aptitude mark' is started as 0 (since it is a number) each time the aptitude rules run. We can then write whatever rules we please to modify it:

```
An aptitude rule:
    if in darkness:
        decrease the aptitude mark by 3.
```

```
An aptitude rule:
    if we have taken the idol:
        increase the aptitude mark by 10.
```

And we had better do something with the result:

```
The last aptitude rule: say "Your aptitude rating is [aptitude mark]."
```

A rulebook can have any number of variables like this. They behave much like "let" values except that they last for a whole rulebook, not an individual rule or To phrase definition. (Well, strictly speaking they are accessible not just to the rules which belong to the rulebook, but also to any rules which previously belonged to the rulebook but were kicked out by means of an explicit rule-listing sentence. This is good because otherwise they will suddenly cause problem messages when unlisted.)

Start of Chapter 19: Rulebooks
Back to §19.9. Basis of a rulebook
Onward to §19.11. Success and failure

§19.11. Success and failure

Though we have blurred over this point so far, each rule must ordinarily end with one of three outcomes: success, failure and neither ("no outcome").

When a rulebook is followed, what happens is that each of its rules is followed in turn **until one of them ends in success or failure** - if ever: it is possible that each rule is tried and each ends with no outcome, so that the rulebook simply runs out of rules to try.

For some rulebooks, these are not useful ideas: "every turn" rules, for instance, by default never produce an outcome, which is why the "every turn" rulebook normally runs through all its rules at the end of each turn. (Use of the phrases below can change that, so it's best not to use them in "every turn" rules.) But for other rulebooks, such as "check taking", it's important that a rule which fails will stop the whole rulebook. For instance, we might find that the "can't take yourself rule" produces no outcome (because we aren't trying to do that), and then likewise the "can't take other people rule" (ditto) but that the "can't take component parts rule" prints up a complaint, and fails the action: the rulebook stops, and never goes on to (for instance) the "can't take scenery rule". This is good, because an impossible action

often fails for several reasons at once, and we only want to print up one objection, not a whole list.

To follow the working of this mechanism, we need to be able to predict the outcome of any given rule. Sometimes this is easy to spot. For instance, in a rule which works on actions:

```
continue the action; means "end this rule with no outcome"
stop the action; means "end this rule in failure"
... instead; means "end this rule in failure"
```

("Success" and "failure" are technical terms here: they do not mean that the player has or hasn't got what he wanted.) This is why the rule:

```
Before taking something: say "The sentry won't let you!" instead.
```

ends in failure, and therefore stops the "before" rulebook. Another easy-to-spot case is when a rule makes use of the explicit phrases:

rule succeeds

This causes the current rule to end immediately, with its outcome considered to be a success. The rulebook being worked through also ends, and is also a success.

rule fails

This causes the current rule to end immediately, with its outcome considered to be a failure. The rulebook being worked through also ends, and is also a failure.

make no decision

This causes the current rule to end immediately, but with no outcome. That means the rulebook being worked through will continue to run on, beginning with the next rule.

But what happens if a rule simply doesn't say whether it succeeds, fails or has no

outcome? In that case **it depends on the rulebook**. For almost all rulebooks, a rule which doesn't make a choice has no outcome, as in the following example:

```
Before taking something: say "The sentry looks at you anxiously!"
```

This rule, if it takes effect, ends with no outcome - so the action continues. But other rulebooks have a different convention: the most important is "instead", where a rule making no explicit choice is deemed to end in failure. For instance:

```
Instead of taking something: say "The sentry prods you with his rifle!"
```

This rule, if it takes effect, ends in failure and therefore stops the action.

We call this the **default outcome** of a rulebook. The default outcome of "before" (and of almost all rulebooks, in fact) is no outcome; the default outcome of "instead" is failure; the default outcome of "after" is success. The few exceptional cases with default outcome success or failure are marked as such in the Rules index.

When we create a rulebook, it will default to "no outcome". But we can specify otherwise with sentences like so:

```
The cosmic analysis rules are a rulebook. The cosmic analysis rules
have default failure.
```

Finally, note that the default outcome for a rulebook is really the default outcome for any rule in that rulebook: if no rules in the rulebook ever apply, for instance if there aren't any and the rulebook is empty, then the rulebook ends with no outcome at all.

We can test the latest outcome like so:

if rule succeeded:

This condition is true if the most recently followed rule or rulebook ended in success. Example:

```
follow the hypothetical clever rule;
if rule succeeded:
    ...
```

if rule failed:

This condition is true if the most recently followed rule or rulebook ended in failure. Example:

```
follow the hypothetical clever rule;
if rule failed:
    ...
```

Note that this is not the opposite of "rule succeeded", because there's a third possibility: that it ended with no outcome.

Start of Chapter 19: Rulebooks

Back to §19.10. Rulebook variables

Onward to §19.12. Named outcomes

Example 404: ★★ **Kyoto** Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

§19.12. Named outcomes

We have seen that the terms "success" and "failure" can be misleading - after all, it might be a good thing for a particular rulebook to "fail". At any rate, these are vague terms, and we don't want to have to remember the conventions used by every rulebook. This is why certain rulebooks have explicitly named outcomes instead. For instance, the "visibility" rules are allowed to have the outcomes:

```
there is sufficient light;
there is insufficient light;
```

These look like phrases, but are in fact named outcomes which can only be used in visibility rules. (They would make no sense elsewhere, and Inform will not allow their use if they are clearly out of context.) Such named outcomes are listed in the Rules

index.

There can be any number of named outcomes. For instance, the Standard Rules define:

```
The does the player mean rules are a rulebook. The does the player
mean rules have outcomes it is very likely, it is likely, it is possible, it
is unlikely and it is very unlikely.
```

which makes five possible outcomes. Five outcomes seems to contradict the principle that there are only three possible outcomes for a rule: in fact, though, the five are counted as five different forms of "success", and any of them will cause a "does the player mean" rule to succeed. If we do not want this, we can instead specify explicitly how the named outcomes correspond to success, failure or "no outcome":

```
Visibility rules have outcomes there is sufficient light (failure) and
there is insufficient light (success).
```

Again, see the Rules index for examples.

The same named outcome can be used for more than one rulebook, and can have different meanings in the context of different rulebooks - "good news" could be defined as success in one rulebook and failure in another, for instance. (This means that rulebook creators need not worry about name clashes and is an important difference in behaviour between rulebook outcomes and kinds of value.) We can even name a specific named outcome as the default outcome for rules in this rulebook:

```
Audibility rules have outcomes high background noise (failure), low
background noise (success - the default) and absolute silence
(success).
```

After a rulebook using named outcomes has run, we can test which outcome occurred by using the phrase:

outcome of the rulebook ... rulebook outcome

This phrase produces the (named) outcome of the phrase most recently followed. Example:

```
follow the audibility rules;
if the outcome of the rulebook is the absolute silence
outcome:
    say "You could hear a pin drop in here."
```

Each named outcome is a value if followed by the word "outcome", which is how "absolute silence" has become "the absolute silence outcome". Named outcomes can be said, so we could use the text substitution "[outcome of the rulebook]", for instance. A final caveat: it is perfectly legal to create a named outcome which means "no outcome", but if so then this will never be "the outcome of the rulebook" because "no outcome" is not an outcome.

Start of Chapter 19: Rulebooks

Back to §19.11. Success and failure

Onward to §19.13. Rulebooks producing values

Example 405: **Being Peter** A set of rules determining the attitude a character will take when asked about certain topics.

---

§19.13. Rulebooks producing values

We have now seen two ways to write the outcome of a rule: as simple success or failure, with more or less explicit phrases like:

```
rule succeeds;
rule fails;
continue the action;
stop the action;
```

and by using a named outcome for the current rulebook as if it were a phrase, as in:

```
low background noise;
```

There is still a third way: we can stop a rule and at the same time produce a value. This isn't needed very often - none of the built-in rulebooks in the Standard Rules produces a value.

As we've seen, every rulebook has one kind of value as its basis, and it also has another kind of value for it to produce. If we call these K and L, then we have altogether four ways to write down the kind of a rulebook:

```
rulebook
K based rulebook
rulebook producing L
K based rulebook producing L
```

If we don't mention K, Inform assumes the rulebook is action based. If we don't mention L, Inform assumes L is "nothing", that is, Inform assumes no value is ever produced. Thus

```
Drum summons rules is a rulebook.
```

is equivalent to

```
Drum summons rules is an action based rulebook producing nothing.
```

But let's now look at a rulebook which does produce something.

```
The cat behavior rules is a rulebook producing an object.
```

This rulebook works out which thing the cat will destroy next. We might have rules like this one:

```
Cat behavior when Austin can see the ball of wool:
    rule succeeds with result the ball of wool.
```

The value is produced only when a rule succeeds, using this phrase:

rule succeeds with result (value)

This phrase can only be used in a rule which produces a value, and the value given must be of the right kind. It causes the current rule to finish immediately, to succeed, and to produce the value given.

How are we to use the cat behavior rulebook? If we write:

```
follow cat behavior
```

then the rulebook runs just as any other rulebook would, but the value produced is lost at the end, which defeats the point. Instead, we might write:

```
Every turn:
    let the destroyed object be the object produced by the cat behavior
rules;
    if the destroyed object is not nothing:
        say "Austin pounces on [the destroyed object] in a flurry.";
        now the destroyed object is nowhere.
```

The key phrase here is

```
object produced by the cat behavior rules
```

which accesses the value this rulebook produces. In general, we write:

(name of kind) **produced by** (rule producing values) **... value**

This phrase is used to follow the named rule, and to collect the resulting value.

(name of kind) **produced by** (values based rule producing values) **for** (value) **... value**

This phrase is used to follow the named rule based on the value given, and to collect the resulting value.

Start of Chapter 19: Rulebooks
Back to §19.12. Named outcomes
Onward to §19.14. Abide by
Example 406: ★ **Feline**

Behavior A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

Example 407: ★★★ **Tilt 2** A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

---

§19.14. Abide by

It often happens that one rule needs to invoke another one. Most of the time, the best way to do this is with "follow":

```
follow the magical mystery tour rule;
```

More often, though, we want not only to invoke another rule, but also to be guided by its advice. For this, we use the otherwise identical phrase:

abide by (rule)

This phrase applies the given rule, and makes its result the result of the present rule. If the rule being abided by succeeds or fails then the original rule also stops, at once and without going on to any further instructions. Example:

```
The omnibus rule:
    abide by the first rule;
    abide by the second rule;
    abide by the third rule;
    abide by the fourth rule.
```

This duplicates the effect of a rulebook of four rules: the "omnibus rule" tries each in turn, and stops as soon as any of them stop.

abide by (values based rule producing values) **for** (value)

This phrase applies the given rule to the given value, and makes its result the result of the present rule. If the rule being abided by succeeds or fails then the original rule also stops, at once and without going on to any further instructions.

Abide might be used in examples like this one:

```
A thing can be fragile or robust.
This is the can't handle fragile things roughly rule: if the noun is
fragile, say "[The noun] is too fragile for such rough handling."
instead.
A check dropping rule: abide by the can't handle fragile things
roughly rule. A check throwing it at rule: abide by the can't handle
fragile things roughly rule.
```

Had we used "follow" instead of "abide by", then in the event of the player typing "drop angel" the text "The glass angel is too fragile for such rough handling" would be printed, which is correct - but then the action would continue as though no difficulty had occurred, which is definitely not correct.

Finally, we can "anonymously abide":

anonymously abide by (rule)

or:
anonymously abide by (values based rule producing values) **for** (value)

This phrase applies the given rule, and makes its result the result of the present rule. If the rule being abided by succeeds or fails then the original rule also stops, at once and without going on to any further instructions. However, the rule deemed to have decided the outcome is the one abided by, not the one doing the abiding.

This is only useful in complicated situations where one rulebook uses another which... and so on. Its effect is exactly the same as "abide", except that the rule deemed to have decided the outcome is the one abided by, not the one doing the abiding. It thus allows a rule or rulebook to act purely as a middle-man, never getting the blame or the credit for what happens. The rule which made the decision is often not very relevant anyway, but it's used as the source of the value "reason the action failed" (see the Advanced Actions chapter).

Start of Chapter 19: Rulebooks
Back to §19.13. Rulebooks producing values
Onward to §19.15. Two rulebooks used internally

§19.15. Two rulebooks used internally

Rulebooks handle almost all of the important tasks which an Inform work of IF must carry out in order to keep play going. We have seen them used in clarifying the player's command, supplying missing ingredients, processing the action to see what should happen, responding, and so on: by this point in the documentation, it must look as if Inform uses rulebooks for everything.

This is nearly true. There is not actually a super-rulebook controlling everything. (Such a super-rulebook would need to repeat itself and never finish, something a rulebook is not allowed to do.) Instead, what happens during play looks like so:

1. Following the "when play begins" rulebook.
2. Repeating:
   2(a). Reading and parsing a command into an action;
   2(b). Following the "action processing" rulebook;
   2(c). Following the "turn sequence" rulebook.

until the story has finished.

3. Following the "when play ends" rulebook.

The command parser occasionally calls on the services of activity rulebooks to help it, but otherwise gets on with its job in ways that we do not "see" as Inform 7 users. The rest of what happens involves rulebooks, and in particular two important beneath-the-surface rulebooks: action processing and turn sequence.

The **action processing rules** are used whenever an action must be tried, by whoever tries it. This usually happens in response to player commands, but not always: it might happen because of a "try...", and it can certainly interrupt an existing action.

The **turn sequence rules** are used at the end of each turn, and include housekeeping as well as timekeeping. They consult the "every turn" rulebook, and advance the time of day, among other useful tasks.

In general, we should only modify the operation of these two crucial rulebooks as a last resort. Play can evidently fall to pieces if they cease to work normally.

Start of Chapter 19: Rulebooks

Back to §19.14. Abide by

Onward to §19.16. The Laws for Sorting Rulebooks

Example 408: ★ **Electrified** Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances.

Example 409: ★ **Timeless** A set of actions which do not take any game time at all.

Example 410: ★★ **Endurance** Giving different actions a range of durations using a time allotment rulebook.

Example 411: ★★ **Escape from the Seraglio** Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

---

§19.16. The Laws for Sorting Rulebooks

Large works created by Inform are heaped high with rules, most of them instead rules, but with a leavening of befores and afters as well. What will happen if these conflict with each other? For instance:

> Instead of opening a container, say "Your mother-in-law looks on with such evident disappointment that you withdraw your hand again."

> Instead of opening an open container, say "Your daughter tuts in

theatrical exasperation at your, like, lameness."

Such clashes are resolved by sorting the rulebooks in order of specificity: thus your daughter gets in before your mother-in-law, because although both have rules hanging on the "opening" action, "an open container" is more specific than "a container". The full set of Laws used by Inform to sort rulebooks is quite elaborate. As we've seen, practical consequences can be investigated using the Rules index; and in most cases, the results are either natural (as above) or irrelevant (because the two rules being compared could not both activate at the same time anyway); but the full set of Laws is laid out below, for reference. It is probably a bad idea to write source text which absolutely relies on non-obvious rule sorting conventions, just the same, because this will make the source text harder to read and understand.

Sorting is done by comparing rules in pairs to decide which is more specific. We shall call these rules X and Y. The Laws are tried in sequence; the first Law to distinguish X and Y gets to decide which is more specific. If no Law is able to decide, X and Y go into the rulebook in order of their appearance in the source text - that is, whichever is defined first appears earlier in the rulebook and therefore takes priority.

Law I - Number of aspects constrained. For action-based rulebooks, rules are scored from 0 to 6 according to whether they constrain any of: (i) the exotic "going" clauses (pushing, by and through), (ii) the location of the action (in, from and to), (iii) the things directly involved (actor, noun, second noun, "nowhere" in the case of "going"), (iv) the presence of others (in the presence of...), (v) the time at which the action occurs (when, or "for the nth time" or "for the nth turn"), and/or (vi) the scene the action occurs in (during). For value based rulebooks, rules are scored from 0 to 3 according to whether they constrain: (i) the value parameter, (ii) the scene in which the rulebook is followed (when, during), and/or (iii) any condition which must hold or activities going on at the same time (when/while). A higher score is more specific than a lower one.

Law II - When/while requirement. A rule with a when/while restriction beats one without.

Law III - Action requirement. A rule with a more specific action requirement beats one with a more general action requirement. (Or similarly, for value based rulebooks, a rule with a more specific parameter requirement beats a more general one.) Details are given below.

Law IV - Scene requirement. A rule with a scene restriction ("during") beats one without.

Details of Law III now follow:

Law III.1 - Object To Which Rule Applies. For value based rulebooks only: the more specific value requirement wins.

Law III.2.1 - Action/Where/Going In Exotic Ways. A more specific combination of "...pushing...", "... by ...", and "... through ..." clauses in a "going" action beats a less specific. (Placing conditions on all three of these clauses beats placing conditions on

any two, which beats any one, which beats none at all.) In cases where X and Y each place, let's say, two such conditions, they are considered in the order "...pushing...", "...by..." and then "...through..." until one wins. (The idea here is that pushing something from room to room is rarer than travelling in a vehicle, which in turn is rarer than going through a door. The rarer action goes first, as more specific.)

Law III.2.2 - Action/Where/Room Where Action Takes Place. A more specific combination of conditions on the room in which the action starts, and in which it ends, beats a less specific. For all actions other than "going", there is no combination to be considered, and what we do is to look at the specificity of the "... in ..." clause. (So "Before looking in the Taj Mahal" beats "Before looking".)

For "going" actions, there are strictly speaking three possible room clauses: "... in ...", "... from ..." and "... to ...". However, "... in ..." and "... from ..." cannot both be present, so that in practice a "going" rule constraining two rooms beats a "going" rule constraining only one.

If both the room gone from (the "...in..." or "...from..." room, whichever is given) and the room gone to (the "... to..." room) are constrained, then the constraints are looked at in the order from-room followed by to-room, since an action which goes to room Z could start in many different places and thus is likely to be more general.

Giving a place as a specific room beats giving only the name of a region; if region R is entirely within region S, then a rule applying in R beats a rule applying in S. (Note that regions can only overlap if one is contained in the other, so this does not lead to ambiguity.)

Law III.2.3 - Action/Where/In The Presence Of. A more specific "...in the presence of..." clause beats a less specific one. (This is again a constraint on where the action can take place, but it's now a potentially a constraint which could be passed in many different places at different times, so it's the most likely to be achieved and therefore the last to be considered of the Laws on Where.)

Law III.3.1 - Action/What/Second Thing Acted On. A more specific constraint on the second noun beats a less specific. Thus "putting something in the wooden box" beats "putting something in a container".

Law III.3.2 - Action/What/Thing Acted On. A more specific constraint on the first noun beats a less specific. Thus "taking a container which is on a supporter" beats "taking a container".

In the case of "going" actions, the first noun is a direction. The special constraint "going nowhere" (which means: a direction in which the actor's location has no map connection) is considered more general than any other constraint placed on the first noun, but more specific than having no constraint at all. Thus "Instead of going north" beats "Instead of going nowhere" which beats "Instead of going".

Law III.3.3 - Action/What/Actor Performing Action. A more specific constraint on the actor beats a less specific.

Law III.4.1 - Action/How/What Happens. A more specific set of actions beats a less specific. For instance, "taking" beats "taking or dropping" beats "doing something other than looking" beats "doing something". A named kind of action (such as "behaving badly") is more specific than "doing something", but considered less specific than any explicitly spelled out list of actions.

Law III.5.1 - Action/When/Duration. An action with a constraint on its history ("for the fifth time", say, or "for the fifth turn") beats one without. If both rules place constraints on history, then the one occurring on the smaller number of possible turns wins (thus "for the third to seventh time" - 5 possible turns of applicability - beats "for less than the tenth turn" - 9 possible turns).

Law III.5.2 - Action/When/Circumstances. A more specific condition under "...when..." beats a less specific one. These conditions could potentially be complex: Inform judges how specific they are by counting the clauses found in them. The more clauses, the more specific the condition, it is assumed.

Law III.6.1 - Action/Name/Is This Named. A rule with a name ("the apple blossom rule", say) beats a rule without.

Start of Chapter 19: Rulebooks

Back to §19.15. Two rulebooks used internally

Onward to Chapter 20: Advanced Text: §20.1. Changing texts

Examples from Chapter 19: Rulebooks

Start of this chapter

Chapter 20: Advanced Text

Indexes of the examples

Example 388 - Nine AM Appointment *RB*

A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

> If there's some reason the player needs to be at a specific place and time, we might want to allow him to wait a number of minutes at once.

```
"Nine AM Appointment"


Waiting more is an action applying to one number.


Understand "wait [a time period]" or "wait for [a time
period]" or "wait for a/an [a time period]" or "wait a/an [a
time period]" as waiting more.


Carry out waiting more:
    let the target time be the time of day plus the time
understood;
    decrease the target time by one minute;
    while the time of day is not the target time:
        follow the turn sequence rules.
```

The one nuance here is that after our wait command occurs, the turn sequence rules will occur one more time. So we need to subtract one minute from the parsed time to make the turn end on the desired number of minutes.

```
Report waiting more:
    say "It is now [time of day + 1 minute]."
```

And if we want to ensure that the player doesn't (accidentally or intentionally) put the interpreter through a really long loop, we could put an upper limit on his patience:

```
Check waiting more:
    if the time understood is greater than one hour, say
"You really haven't got that kind of patience." instead.


The Specialist's Office is a room. The secretary is a
woman in the Office. Instead of asking the secretary
about "[appointment]", say "'Hang on just five more
minutes,' she says, in a distracted manner."
```

```
Understand "appointment" or "specialist" or "doctor" as
"[appointment]".


At 9:45 AM: say "At [the time of day in words], secretary
glances at you and gives a reassuring smile."


Test me with "ask secretary about appointment / wait
five minutes / g / g / wait 61 minutes / wait for half an
hour / wait for a quarter of an hour / wait for an hour".
```

---

389 **Example** **Delayed Gratification** *RB*

A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

```
"Delayed Gratification"


Hanging around until is an action applying to one time.


Check hanging around until:
    if the time of day is the time understood, say "It is
[time understood] now!" instead;
    if the time of day is after the time understood, say "It
is too late for that now." instead.


Carry out hanging around until:
    while the time of day is before the time understood:
        follow the turn sequence rules.
```

```
Report hanging around until:
    say "You yawn until [time understood]."

Understand "wait until [time]" as hanging around until.

The Empty Field is a room. "It's an ordinary empty field.
Nothing to see here at all-- yet. Wait until 11:45 PM,
though."

At 11:45 PM:
    say "Suddenly the air is filled with light and the
sounds of an approaching band. Over the crest of the
hill comes a parade of singing, stomping, hooting
people: and not just people, but dogs, horses,
elephants, giraffes... There are banners, and candles,
and a flag that glows eerie-green in the dark; there is a
float shaped like an enormous turtle, its shell covered
with winking green lights; there is an old man dressed
as a skeleton, carried in a litter, his neck garlanded with
dried chiles. There are small girls throwing rose petals
from a basket, and grown women half-naked carrying
the emblems of Bacchic revelry, and two little boys each
with a silver basin of clear water. All these go by in
procession, and you join on at the end.";
    end the story finally.

Test me with "look / z / z / wait until 11:45 PM".
```

Example The Crane's Leg 2 *RB*

A description text generated based on the propensities of the player-character, following different rulebooks for different characters.

Names of rules can be listed in tables. This is convenient if, for instance, we decide that we'd like to swap the rules we use for a specific purpose, as in this continuation of our earlier example of automated description:

"The Crane's Leg, Grown Longer"

Material is a kind of value. The materials are wood, glass, stone, cloth, paper, clay, and metal. A thing has a material.

Color is a kind of value. The colors are red, orange, yellow, green, blue, indigo, violet, black, brown, and white. A thing has a color. A thing is usually white.

A height is a kind of value. 3 feet 11 inches specifies a height. A thing has a height. Definition: a thing is tall if its height is 6 feet 0 inches or more. Definition: a thing is short if its height is 2 feet 0 inches or less.

Imitation relates various things to one thing (called the ideal). The verb to imitate means the imitation relation.

A table is a kind of supporter. A table is usually wood. The height of a table is usually 3 feet 8 inches. The ordinary table is a table. Every table imitates the ordinary table.

A rock is a kind of thing. A rock is usually stone. The ordinary rock is a rock. The height of a rock is usually 0 feet 3 inches. Every rock imitates the ordinary rock.

The description of a thing is usually "[comparison with ideal][run paragraph on]".

```
To say comparison with ideal:
    say "You observe [the noun]:[paragraph break]";
    choose row with character of the player in Table of
Descriptive Reporting;
    follow instructions entry.



This is the comparative observation rule:
    let the sample be the ideal of the noun;
    if the sample is not a thing:
        say "Nothing special, really.";
        rule succeeds;
    if the material of the noun is not the material of the
sample:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun], and made of
[material of the noun].";
            otherwise say "Unusually tall at [height of the
noun], and made of [material of the noun].";
        otherwise:
            say "Distinct mostly in being made of [material of
the noun].";
    otherwise:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun].";
            otherwise say "Unusually tall at [height of the
noun].";
        otherwise:
            say "In every respect [a sample]."



The Pleasure Garden is a room. "At the riverbank, a
pleasing garden, having many curving paths and one
straight."



The low table is a table in the Pleasure Garden. The
height of the low table is 2 feet 3 inches. On the low
table is a yellow metal rock called a gold nugget. A
willow is in the Pleasure Garden. The height of the
willow is 20 feet 2 inches.
```

```
Understand "possess [any person]" or "be [any person]"
as possessing.


Possessing is an action applying to one thing. Carry out
possessing: now the player is the noun; say "You swap
bodies!"


The crane is a person in the Garden. The height of the
crane is 4 feet 0 inches.


Table of Descriptive Reporting

character descriptions
the character comparative observation rule
yourself the observation rule
bird

This is the bird observation rule:
    if the noun is shorter than the player, say "Small, like
a duck[if the color of the noun is not white]; and [color of
the noun][end if].";
    otherwise say "Supremely tall[if the color of the noun
is not white] and [color of the noun][end if]."


Test me with "examine table / examine nugget /
examine willow / possess crane / examine table /
examine nugget / examine willow".
```

ExampleStone *RB*

A soup to which the player can add ingredients, which will have different effects when the player eats.

A thing can have a rule as a property, if we like. Here we are going to allow the player to make a soup whose effects will depend on its ingredients. Each ingredient will have its own "food effect" rule, to be followed when the food is eaten.

Note that there are other, slightly less cumbersome ways to do the same thing -- we will see in the chapter on Rulebooks that we could make a "food effects rulebook" and then write a number of rules such as "food effects rule for carrots" or "food effects rule for the stone". Nonetheless, we demonstrate rules-as-properties here for the sake of thoroughness.

So:

```
"Stone"


A food is a kind of thing that is edible. A food has a rule
called the food effect.


Carry out eating a food:
    if a food is part of the noun:
        repeat with item running through things which are
part of the noun:
            if item is a food, follow the food effect of the
item;
    follow the food effect of the noun.


Report eating a food:
    say "You eat [the noun]. [diagnosis of the player]";
    stop the action.


To say diagnosis of (victim - a person):
    if the victim is ill:
        say "You are ill.";
        rule succeeds;
    otherwise:
        say "You are healthy. ";
    if the victim is awake, say "You are wide awake. ";
    otherwise say "You are sleepy. ";
    if the victim is bright-eyed, say "Your eyesight is
```

```
clear. ";
    otherwise say "Your eyesight is dim. ";
    if the victim is weak, say "You are weak. ";
    otherwise say "You are strong. ";
    if the victim is hungry, say "You are hungry.";
    otherwise say "You are well-fed."
```

And now to provide some particular foods:

```
Some carrots are a food. The food effect of carrots is
the bright-eye rule. This is the bright-eye rule: now the
player is bright-eyed.
```

```
Some potatoes are a food. The food effect of the
potatoes is the sleepiness rule. This is the sleepiness
rule: now the player is sleepy.
```

```
The broth is a food. The indefinite article of the broth is
"some". The food effect of broth is the filling rule. This is
the filling rule: now the player is full.
```

```
The hambone is a food. The food effect of the hambone
is the heartiness rule. This is the heartiness rule: now
the player is strong. Instead of eating the hambone: say
"You cannot just eat a bone!"
```

```
The poison ivy is a food. "Poison ivy grows nearby." The
food effect of poison is the illness rule. This is the illness
rule: now the player is ill.
```

```
A person can be bright-eyed or blind. The player is
blind.
```

```
A person can be well or ill. The player is well.
```

```
A person can be hungry or full. The player is full.

A person can be strong or weak. The player is weak.

A person can be awake or sleepy. The player is sleepy.

The broth is in the kettle. The kettle is on the fire. The
fire is in the Clearing. The Clearing is a room.

The player carries the hambone, the potatoes, and the
carrots. The ivy is in the clearing.

Instead of examining the broth:
    if something is part of the broth, say "In the broth, [a
list of things that are part of the broth] float[if exactly one
thing is part of the broth]s[end if].";
    otherwise say "It is just a thin broth with no other
ingredients."

Instead of inserting something into the broth: try
inserting the noun into the holder of the broth.

Instead of taking the broth: say "You cannot take the
broth in your bare hands."
```

And the following is a relatively unimportant nicety:

```
To sink is a verb.
```

```
After inserting a food which is not the broth into a
container which contains the broth:
    now the noun is part of the broth;
    say "[The noun] [sink] into [the second noun], making
the broth richer."


Test me with "x broth / eat hambone / put hambone in
kettle / x broth / put potatoes in broth / x broth / eat
carrots / eat broth / put ivy in kettle / eat ivy".
```

---

392 **ExampleBribery** *RB*

A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

If we want to rewrite the functionality of a command that usually ends with a "block..." rule, we will have to begin by turning the blocking off.

```
"Bribery"


The block giving rule is not listed in the check giving it to
rules.
```

As it happens, correct behavior is built into the GIVE command once "block giving" is turned off, so we do not have to write a replacement report or carry-out rule; the object will be transferred to the possession of the caterpillar. But we do want to adjust the action just a little so that our gift cheers up the recipient:

```
Carry out giving (this is the gratitude for gifts rule):
improve the mood of the second noun.
```

```
Mood is a kind of value. The moods are hostile,
suspicious, indifferent, friendly, and adoring. An animal
has a mood. An animal is usually indifferent.


To improve the mood of (character - an animal):
    if the mood of character is less than friendly, now the
mood of the character is the mood after the mood of the
character.
```

Now whenever we give something to an animal, the animal will be pleased about the present. Of course, we might also want to add a check rule to giving, to see whether the offering is something the recipient really wants:

```
Check giving (this is the polite refusal of unwanted
objects rule):
    unless the noun interests the second noun:
        say "[The second noun] disdainfully refuses [the
noun]." instead.


To decide whether (item - a thing) interests (character -
a person):
    if the character has the item, no;
    if the item is edible, yes;
    no.


Instead of showing something to someone:
try giving the noun to the second noun.
```

There is already a perfectly workable report rule that will describe what happens when we give something to someone, but let's say we want to report on the recipient's changed mood, too:

```
After giving something to someone:
    say "You give [the noun] to [the second noun], who
appears mollified and is now merely [mood of the
second noun]."
```

And the rest is all scenario:

The Leafy Branch is a room. "You stand on smooth bark dappled by sunlight. The scent-trail runs forward to home.

The branch continues forward and backward from here, and a stem extends forward-up."

Instead of going south in Leafy Branch, say "You must not back down! The scent trail leads onward!"

The Very Hungry Caterpillar is a hostile animal in the Leafy Branch. "[The Caterpillar] looks [mood]." Instead of examining the Caterpillar, say "[The caterpillar] appears [mood]."

The player carries an edible thing called a peanut crumb. The carrying capacity of the player is 1. After taking something, say "You lift [the noun], though it is nearly your own size."

Instead of going north in the presence of a hostile caterpillar:
    say "[The Caterpillar] moves to block your exit, glaring down at you with all the bristles on its skin extended to full size."

Instead of going north in the presence of a suspicious caterpillar:
    say "[The Caterpillar] moves to block your exit, though it might allow you past if you offered further tribute."

```
The Leaf Face is above the branch. "The smooth and
shiny surface of the leaf extends forward from here, but
you have lost the scent-trail. This is not the way home."
The pear fragment is an edible thing in Leaf Face. The
dead aphid is a thing in Leaf Face.


The Twig is north of Leafy Branch. "The scent-trail is
weak but not entirely gone, and you pursue it
faithfully..."


After going to the Twig:
    say "The scent-trail is weak but not entirely gone, and
you pursue it faithfully...";
    end the story finally.


Understand "forward-up" as up. Understand "forward"
as north. Understand "backward" as south. Understand
"backward-down" as down.


Test me with "forward / give crumb to caterpillar /
forward / forward-up / get aphid / get fragment / down /
give aphid to caterpillar / drop aphid / forward-up / get
fragment / down / give fragment / forward".
```

---

393 **Example** **Saint Eligius** *RB*

Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

A not-infrequent desire in IF is to provide a few lines of comment when the player first enters a new room, after the objects are described but before anything else (such as an every turn rule) can fire. The cleanest, most systematic solution is to add a rule to the carry out looking rulebook, so:

"Saint Eligius"

The first look rule is listed after the room description paragraphs about objects rule in the carry out looking rules. A room can be commented or uncommented. A room is usually uncommented.

This is the first look rule:
    if the location is uncommented, carry out the gawking at activity with the location.

Gawking at something is an activity.

Rule for gawking at the Diamond Market:
    say "Your throat closes and your eyes begin to sting. You have long disdained pomp and luxury, and railed against the oppression that brings such wealth to some men at the cost of the lives of others; you were not prepared for the magnificence."

After gawking at a room (called the target): now the target is commented.

And now the scene itself:

The Cobbled Alley is a room. "The Alley has never been made into a proper street: the buildings on either side are simply too important to tear down. For all that, there isn't much sign of the magnificence nearby. The entrance you seek is set below street level, four grimy steps down to a half-basement."

After going to Diamond Market:
    say "You descend the steps quickly and step into the

small foyer, allowing yourself to be searched for weapons, before going on to...";
    continue the action.

Diamond Market is down from Cobbled Alley. "The roof is vaulted and painted in allegorical images representing Plenty, the Riches of the Earth, and Saint Eligius, patron of goldsmiths and jewelers.

Under their watchful eye, dozens of men in sober black robes sit; and on the tables before them are rubies, emeralds, sapphires from oversea, but most of all diamonds, both raw and cut."

The burly guard is a man in Diamond Market. "A burly guard patrols quite close to you, but even he is more sumptuously dressed than the average burly guard, and his buttons shine."

Test me with "d / look".

Example **Uptempo**

RB

Adjust time advancement so the game clock moves fifteen minutes each turn.

Suppose a game in which all actions take a very long time. Here's a simple implementation:

"Uptempo"

The fast time rule is listed instead of the advance time rule in the turn sequence rules.

```
This is the fast time rule:
    increment the turn count;
    increase the time of day by 15 minutes.


When play begins: now the right hand status line is
"[time of day]".


The Temporal Hot Spot is a room.


Test me with "z / z".
```

This works fine as it stands, but we may run into some difficulty with it if we add scheduled events:

```
At 9:30 AM:
    say "Two turtles run by, almost too fast to see."


At 9:37 AM:
    say "A snail blitzes past."


At 9:42 AM:
    say "The grass grows."


At 9:50 AM:
    say "Several flowers burst open."
```

Time is counted forward after the schedule has already been consulted, so that only the 9:30 AM event happens between 9:30 and 9:45; the next two appear to occur between 9:45 and 10:00 AM, and the 9:50 AM event is not reported until the 10:00 AM to 10:15 wait. To get around this, we might schedule events only on the fifteen-minute mark when we want

them to occur. Alternatively, we might try instead

```
"Uptempo"

The fast time rule is listed before the timed events rule
in the turn sequence rules.

The advance time rule is not listed in the turn sequence
rules.

This is the fast time rule:
    increment the turn count;
    increase the time of day by 15 minutes.

When play begins: now the right hand status line is
"[time of day]".

The Temporal Hot Spot is a room.

At 9:30 AM:
    say "Two turtles run by, almost too fast to see."

At 9:37 AM:
    say "A snail blitzes past."

At 9:42 AM:
    say "The grass grows."
```

```
At 9:50 AM:
    say "Several flowers burst open."


Test me with "z / z / z / z".
```

This time our revised time-advancing rule is listed just before the event scheduler, not just afterwards.

---

395 Example Verbosity 2 *RB*

Making rooms give full descriptions each time we enter, even if we have visited before, and disallowing player use of BRIEF and SUPERBRIEF.

Suppose that we want the player always to see full room descriptions, even if he tries to reset the defaults -- perhaps because there is vital information there which he will miss if he turns off full-length room descriptions.

To do this, we might want to remove the standard behavior of the three actions associated with BRIEF, SUPERBRIEF, and VERBOSE, replacing them with explanatory messages about how the game behaves. We cannot use Instead to override these actions, because Instead rules do not apply to actions out of world. Instead, we will want to remove and replace the carry out rules.

We can do this easily by going to the Actions Index, looking up the detail panel for, say, "preferring abbreviated room descriptions", and click the "unlist" button to paste in the sentence that will remove this rule from the rulebook.

Let's remove all three of the carry out rules and substitute our own:

```
"Verbosity 2"


Section 1 - Procedure


The prefer unabbreviated room descriptions rule is not
```

listed in the carry out preferring unabbreviated room descriptions rulebook.

The prefer sometimes abbreviated room descriptions rule is not listed in the carry out preferring sometimes abbreviated room descriptions rulebook.

The prefer abbreviated room descriptions rule is not listed in the carry out preferring abbreviated room descriptions rulebook.

Carry out preferring unabbreviated room descriptions:
    say "[story title] always provides full-length descriptions for your reading pleasure."

Carry out preferring sometimes abbreviated room descriptions:
    say "For your playing protection, [story title] provides only full-length room descriptions."

Carry out preferring abbreviated room descriptions:
    try preferring sometimes abbreviated room descriptions instead.

The standard report preferring abbreviated room descriptions rule is not listed in the report preferring abbreviated room descriptions rulebook.

The standard report preferring unabbreviated room descriptions rule is not listed in the report preferring unabbreviated room descriptions rulebook.

The standard report preferring sometimes abbreviated room descriptions rule is not listed in the report preferring sometimes abbreviated room descriptions rulebook.

Use full-length room descriptions.

Section 2 - Scenario

The Wilkie Memorial Research Wing is a room. "The research wing was built onto the science building in 1967, when the college's finances were good but its aesthetic standards at a local minimum. A dull brown corridor recedes both north and south; drab olive doors open onto the laboratories of individual faculty members. The twitchy fluorescent lighting makes the whole thing flicker, as though it might wink out of existence at any moment.

The Men's Restroom is immediately west of this point."

The Men's Restroom is west of the Research Wing. "Well, yes, you really shouldn't be in here. But the nearest women's room is on the other side of the building, and at this hour you have the labs mostly to yourself. All the same, you try not to read any of the things scrawled over the urinals which might have been intended in confidence."

Test me with "west / east / brief / w / e / superbrief / w / e / verbose".

A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

Suppose we want to let the player explicitly sit, stand, or lie down on different enterable objects. (Normally Inform treats all these actions as entering, but there may be cases where we want to distinguish between the player sitting on a chair and the player standing on it.)

Our implementation gives each kind of enterable a range of allowed postures, and one preferred posture. If the player enters the supporter or container without specifying a posture (as in ENTER CHAIR), he will be put in the preferred posture. If he explicitly says, e.g., STAND ON CHAIR, he will be put in the standing position as long as standing is a posture that suits the chair.

```
"Slouching"

Section 1 - Posture Rules

A posture is a kind of value. The postures are seated,
standing, and reclining.

A person has a posture. The posture of a person is
usually standing.

A supporter has a posture. A container has a posture.

Posture-permission relates various things to various
postures. The verb to allow means the posture-
permission relation.
```

Understand the commands "stand" and "sit" and "lie" as something new.

Understand "sit on/in [something]" as sitting on.
Understand "lie on/in [something]" as lying on.
Understand "stand on/in [something]" as standing up on.

Sitting on is an action applying to one thing.
Lying on is an action applying to one thing.
Standing up on is an action applying to one thing.

Carry out an actor sitting on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not seated, try the actor taking position seated;
        otherwise follow the report taking position rules.

Carry out an actor lying on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not reclining, try the actor taking position reclining;
        otherwise follow the report taking position rules.

Carry out an actor standing up on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not standing, try the actor taking position standing;
        otherwise follow the report taking position rules.

Understand "lie down" as lying down.

```
Understand "sit down" or "sit" or "sit up" as sitting down.
Understand "stand" or "stand up" as standing up.


Lying down is an action applying to nothing.
Sitting down is an action applying to nothing.
Standing up is an action applying to nothing.


Taking position is an action applying to one posture.


Instead of an actor lying down:
    try the actor taking position reclining; rule succeeds.
Instead of an actor sitting down:
    try the actor taking position seated; rule succeeds.
Instead of an actor standing up:
    try the actor taking position standing; rule succeeds.


Check an actor taking position:
    if the holder of the actor is not a room and the holder
of the actor does not allow the posture understood:
        if the actor is the player:
            say "You can't take that position [in-on the
holder of the actor].";
        otherwise if the actor is visible:
            say "[The actor] can't take that position.";
        stop the action.


Check an actor taking position:
    if the posture understood is the posture of the actor:
        if the actor is the player:
            say "You are already [the posture understood].";
        otherwise:
            if the actor is visible, say "[The actor] is already
[the posture understood].";
        stop the action.


Carry out an actor taking position:
```

```
    now the posture of the actor is the posture
understood.


Report someone taking position (this is the position-
report rule):
    say "[The actor] is now [the posture of the actor][if the
holder of the actor is not the location of the actor] [in-on
the holder of the actor][end if]."


Report taking position:
    say "You are now [the posture of the player][if the
holder of the player is not the location] [in-on the holder
of the player][end if]."


To say in-on (item - a thing):
    if the item is a container, say "in [the item]";
    otherwise say "on [the item]".


Carry out an actor exiting (this is the departure-posture
rule):
    let N be the holder of the actor;
    if N is a container or N is a supporter,
        now the posture of the actor is the posture of N;
    otherwise now the posture of the actor is standing.


The departure-posture rule is listed after the standard
exiting rule in the carry out exiting rulebook.
The departure-posture rule is listed after the standard
getting off rule in the carry out getting off rulebook.


Carry out an actor entering something (this is the arrival-
posture rule):
    if the noun is a container or the noun is a supporter,
        now the posture of the actor is the posture of the
noun.
```

The arrival-posture rule is listed after the standard entering rule in the carry out entering rulebook.

```
Check an actor going somewhere:
    if the actor is in a room and the actor is not standing:
        say "([if the actor is not the player][the actor] [end
if]first standing up)[command clarification break]";
        silently try the actor taking position standing;
        if the actor is not standing, stop the action.
```

Section 2 - Some Generic Kinds

A chair is a kind of supporter. A chair is always enterable. Every chair allows seated and standing. A chair is usually seated.
A sofa is a kind of supporter. A sofa is always enterable. Every sofa allows seated, standing and reclining. A sofa is usually seated.
A hammock is a kind of container. A hammock is always enterable. Every hammock allows seated and reclining. A hammock is usually reclining.

Section 3 - The Scenario

The Resort is a room.

The banana hammock is a hammock in the Resort. The stone bench is a sofa in the resort.

Clark is a man in the Resort. A persuasion rule: persuasion succeeds.

```
Rule for writing a paragraph about someone (called
target):
    say "[The target] is [posture] [if the holder of the
target is the location]nearby[otherwise][in-on the holder
of the target][end if]."


Rule for writing a paragraph about something which
encloses an unmentioned person (called target):
    carry out the writing a paragraph about activity with
the target instead.


Test me with "sit on bench / stand on bench / get up / lie
on hammock / sit up / g / clark, sit on bench / look /
clark, lie down / g / look / clark, get up / look / clark, lie
down / look / enter bench".
```

---

397 **ExampleSwigmore U.** *RB*

Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

Inform's default assumption is that if a player on an enterable object drops something, the dropped article winds up beside him on the same supporter or in the same container. This makes lots of sense for a dais, say, or a king-sized bed. It's a little less sensible if the enterable supporter in question is a bar stool or the like. So suppose we want to add a new kind of supporter called a perch, where everything dropped lands on the floor.

There are actually several ways of implementing this, but one of them is to reach right into the drop action and replace the "standard dropping rule" with a different one of our own invention -- like this:

```
"Swigmore U."
```

```
Moe's Tavern is a room. The bar is an enterable
supporter in Moe's. A drink is a kind of thing. On the bar
is a drink called a flaming Homer.


A perch is a kind of supporter. A perch is always
enterable. The stool is a perch in Moe's.


The player carries a dead field mouse and a tomacco
fruit.


The sophisticated dropping rule is listed instead of the
standard dropping rule in the carry out dropping
rulebook.


This is the sophisticated dropping rule:
    if the player is on a perch (called the awkward
position):
        let place be the holder of the awkward position;
        move the noun to the place;
    otherwise:
        move the noun to the holder of the player.


Test me with "sit on stool / drop mouse / look / get up /
look".
```

Now the carry-out behavior of the dropping action has been changed, but we haven't had to interfere in the checks or reporting at all. The rest of the action works just as it always did.

Of course, maybe we do want to change the way the action is reported, to make it clearer to the player where the dropped article wound up:

```
The sophisticated report dropping rule is listed instead
of the standard report dropping rule in the report
dropping rulebook.
```

```
This is the sophisticated report dropping rule:
    say "You drop [the noun] on [if the holder of the noun
is a room]the ground[otherwise][the holder of the noun]
[end if]."
```

Example **Access All Areas** *RB*

The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

If somebody tries to walk through a closed door, the "can't go through closed doors rule" usually stops them. This is a rule belonging to the "check going" rulebook. These names are fairly explanatory when written out, but hard to remember: fortunately we don't need to remember them, as the Index panel contains a full inventory of the check, carry out and report rules for every action, showing all of their names and the order in which they are checked. (We can also find out which rules are stopping an action by typing the testing command ACTIONS.)

Here we make the rule do nothing provided a condition holds:

```
"Access All Areas"

The extremely difficult door is north of the Standing
Room and south of the Room of Walking Upside Down.
It is a locked door.

The player is carrying the Pointy Hat of Liminal
Transgression. The hat is wearable.

The can't go through closed doors rule does nothing
when the Hat is worn.
```

Test me with "n / wear hat / n".

(The Pointy Hat may be useful in debugging a game, even if it never makes it into the final published work.)

---

399 **Example** **Backus-Naur form for rules** *RB*

The full grammar Inform uses to parse rule definitions, in a standard computer-science notation.

Backus-Naur form, or BNF, is a standard notation used by computer scientists to specify more or less precisely what the valid programs are for a given programming language. It tends to provide a good description for a language such as C or Pascal, where contextual rules are limited, but the authors of Inform are doubtful that it is such a good tool for a natural-language system. For those who are interested, though, the following gives a formal specification for Inform's rules.

```
<rule> ::=
    Definition : A/an <kind> is <new adjectival name> if/
unless <definition>
    | <preamble> : <phrases>
    | <preamble> , <phrase> (* only allowed for a few
cases: see below)



<definition> ::=
    <condition>
    | its/his/her/their <value property name> is/are
<value> or less/more
    | : <phrases>



<preamble> ::=
    To <phrase template>
    | To decide if/whether <phrase template>
    | To decide which/what <kind of value> is <phrase
template>
    | This is the <rule name>
    | [[A] Rule for] <circumstances> [(this is the <rule
name>)]
```

```
<circumstances> ::=
    At <time>
    | When <event name>
    | [<placement>] <rulebook reference> [while/when
<condition>] [during <scene name>]



<rulebook reference> ::=
    <rulebook name> [about/for/of/on/rule] [<action
pattern>]
    | <object-based-rulebook name> [about/for/of/on/rule]
[<description>]



<placement> ::=
    a/an
    | [the] first
    | [the] last



<phrases> ::=
    <phrase>
    | <phrases> ; <phrase>
```

The following examples show how Inform breaks down some typical rules using the system above:

```
<rule> = At 2:09 PM: increase the score by 2; say
"Progress!"
    <preamble> = At 2:09 PM
        <circumstances> = At 2:09 PM
            At
            <time> = 2:09 PM
    :
    <phrases> = increase the score by 2; say "Progress!"
        <phrase> = increase the score by 2
        ;
        <phrase> = say "Progress"



<rule> = Instead of eating the ostrich during Formal
```

```
Dinner (this is the cuisine rule), say "It's greasy!"
   <preamble> = Instead of eating the ostrich during
Formal Dinner (this is the cuisine rule)
      <circumstances> = Instead of eating the ostrich
during Formal Dinner
         <rulebook reference> = Instead of eating the
ostrich
            <rulebook name> = Instead
            of
            <action pattern> = eating the ostrich
         during
         <scene name> = Formal Dinner
      (
      this
      is
      the
      <rule name> = cuisine rule
      )
   ,
   <phrases> = say "It's greasy!"
      <phrase> = say "It's greasy!"



<rule> = After printing the name of a container: say "!"
   <preamble> = After printing the name of a container
      <circumstances> = After printing the name of a
container
         <rulebook reference> = After printing the name
of a container
            <object-based-rulebook name> = After
printing the name
            of
            <description> = a container
   :
   <phrases> = say "!"
      <phrase> = say "!"
```

(*) The colon dividing a rule preamble from its definition can be replaced by a comma only if the preamble begins with the words "Instead of", "Before", "After", "Every turn" or "When", and if the definition consists only of a single phrase.

Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the

scenery.

Observation of novice IF players suggests that they often have a hard time figuring out how to get started, especially if they are encountering the game in a context where they don't have time to settle in and read instructions. Here we provide some training wheels to help them learn to communicate.

This is divided into several parts. The first part is the system of rules for general guidance, which could be excerpted and used anywhere. The second part is a scenario using these rules.

```
"Solitude"
```

```
Part 1 - General Rules
```

```
When play begins:
    say "Have you played interactive fiction before? >";
    now novice mode is whether or not the player
consents.
```

The rationale for asking the question this way, and not another, is that novices asked whether they would like instructions very often say no, even if they need them.

```
Novice mode is a truth state that varies. Novice mode is
true.
```

```
Stopping novice mode is an action out of world.
Starting novice mode is an action out of world.
```

```
Understand "novice mode off" or "novice off" as
stopping novice mode.
Understand "novice mode on" or "novice on" as starting
novice mode.
```

```
Carry out stopping novice mode: now novice mode is
false.
Carry out starting novice mode: now novice mode is
true.


Report stopping novice mode: say "Novice mode is now
off."
Report starting novice mode: say "Novice mode is now
on."


Before reading a command when novice mode is true:
    say "[line break]Some options to try:[line break]";
    follow the novice suggestion rules.


The novice suggestion rules is a rulebook.


A novice suggestion rule (this is the suggestion that he
look rule):
    if not looking and not going, say " [bold
type]look[roman type]".


A novice suggestion rule (this is the suggestion that he
check inventory rule):
    if the player carries something and we are not taking
inventory, say " [bold type]inventory[roman type] (I)".


A novice suggestion rule (this is the suggestion that he
put things on rule):
    if the player carries something and a free-standing
supporter is relevant, say " [bold type]put[roman type]
something [bold type]on[roman type] [the list of relevant
supporters]".
```

A novice suggestion rule (this is the suggestion that he take things rule):
    if a gettable thing is relevant, say " [bold type]take[roman type] [the list of gettable relevant things]".

A novice suggestion rule (this is the suggestion that he examine things rule):
    if an unexamined thing is relevant, say " [bold type]examine[roman type] (X) [the list of unexamined relevant things]".

A novice suggestion rule (this is the suggestion that he enter things rule):
    if a relevant thing is worth entering, say " [bold type]enter[roman type] [the list of worth entering relevant things], or [bold type]get out[roman type]".

A novice suggestion rule (this is the suggestion that he open things rule):
    if an unlocked openable thing is relevant, say " [bold type]open[roman type] or [bold type]close[roman type] [the list of unlocked openable relevant things]".

A novice suggestion rule (this is the suggestion that he lock things rule):
    if a closed lockable thing is relevant, say " [bold type]lock[roman type] or [bold type]unlock[roman type] [the list of closed lockable relevant things]".

A novice suggestion rule (this is the suggestion that he eat things rule):
    if the player carries an edible relevant thing, say " [bold type]eat[roman type] [the list of edible relevant things carried by the player]".

A novice suggestion rule (this is the suggestion that he wear things rule):
    if the player carries a wearable relevant thing, say " [bold type]wear[roman type] [the list of wearable relevant things carried by the player]".

A novice suggestion rule (this is the suggestion that he turn things on rule):
    if a device is relevant, say " [bold type]turn on[roman type] or [bold type]turn off[roman type] [the list of relevant devices]".

A novice suggestion rule (this is the suggestion that he go places rule):
    if a room is adjacent, say " [bold type]go[roman type] [exit list][if in darkness] or try other directions in the dark[otherwise]".

A novice suggestion rule (this is the suggestion that he enter doors rule):
    if an open door is relevant, say " [bold type]go through[roman type] [the list of relevant open doors]".

A novice suggestion rule (this is the suggestion that he interact with people rule):
    if another person is relevant, say " [bold type]kiss[roman type] or [bold type]wake[roman type] [the list of relevant other people][if the player carries something], or [bold type]give[roman type] things [bold type]to[roman type] someone[end if]".

A novice suggestion rule (this is the suggestion that he ask for help rule):
    say " [bold type]help[roman type] to see a more complete set of instructions".

```
A novice suggestion rule (this is the suggestion that he
turn off help rule):
    say " [bold type]novice mode off[roman type] to turn
off this guidance".


Last novice suggestion rule:
    say "[line break]".
```

The suggestion about asking for help is no good unless we provide some. This might take any of a number of forms, but for the sake of example we'll use an easy way out:

```
Include Basic Screen Effects by Emily Short. Include
Menus by Emily Short. Include Basic Help Menu by
Emily Short.


After taking inventory when novice mode is true: say "To
get rid of any of these objects, [bold type]drop[roman
type] it."


A thing can be examined or unexamined. Carry out
examining something: now the noun is examined.


A thing can be seen or unseen. A thing is usually
unseen.


Definition: a thing is relevant if it is seen and it is visible.
Before printing the name of something (called the
target): now the target is seen; if novice mode is true,
say "[bold type]". After printing the name of something:
say "[roman type]".


Definition: a supporter is worth entering:
    if the player carries it, no;
```

```
    if it is enterable, yes.


Definition: a container is worth entering:
    if the player carries it, no;
    if it is enterable and it is open, yes.


Definition: a person is other if it is not the player.
Definition: a person is another if it is other.


Definition: a thing is free-standing if it is in a room.


To say exit list:
    let place be location;
    let count be 0;
    repeat with way running through directions:
        let place be the room way from the location;
        if place is a room:
            increment count;
            say "[if count is greater than 1] or[end if] [bold
type][way][roman type]".


Definition: a thing is gettable:
    if it is scenery, no;
    if it is fixed in place, no;
    if it is a person, no;
    if the player is carrying it, no;
    if the player is wearing it, no;
    yes.


Part 2 - On the Ground


Antarctic Research Station is a room. "Though not
always the most stimulating of environments, the station
is far from your ex-wife and most of the things in the
```

world that annoy you, namely the other 6+ billion people. There is a second room to the south." The station contains a radio. The radio is a device. It is fixed in place.

South of the Station is Sitting Room. The description of the Sitting Room is "Just big enough for a very [comfortable chair]." The Sitting Room contains an enterable supporter called a comfortable chair. The chair is scenery. A monograph about penguins is in the Sitting Room.

Blistering Cold is a room. "It is white out here and very very very cold." The white door is a door. "[The white door] leads to [the other side of the white door]." It is west of the Blistering Cold and east of the Antarctic Research Station.

Test me with "i / x radio / x door / s / i / x chair / x monograph / sit in chair / get up / n / open door / enter door".

---

401 **Example** **In Fire or in Flood** *RB*

A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

"In Fire or in Flood"

Part I - Rules for combustion

Heat is a kind of value. The heats are whole, damp, and flaming. A thing has a heat. A thing is usually whole.

A thing has a number called endurance. The endurance of a thing is usually 5. A thing has a number called turns of burning. A thing can be flammable or flame-retardant.

Before printing the name of something flaming:
    say "flaming ".

Before burning something when the player is not carrying something flaming:
    if a flaming portable thing (called the lighter) is touchable:
        say "(with [the lighter], which you first take) [command clarification break]";
        try taking the lighter.

Instead of burning something when the player is not carrying something flaming:
    say "You would first need a fire source."

Instead of burning something flame-retardant:
    say "[The noun] is not the sort of thing that catches fire."

Instead of burning something flammable when the player is carrying something flaming (called the flame source):
    say "You light [the noun] with [the flame source].";
    now the heat of the noun is flaming.

Instead of burning something when the player is in the noun:

```
    say "That seems dangerous given that you yourself
are in [the noun]."


Instead of burning something when the player is on the
noun:
    say "That seems dangerous given that you yourself
are on [the noun]."


Instead of examining something:
    say "Hm, the [printed name] appears to be [heat]."


Before taking a flaming thing:
    let turns remaining be the endurance of the noun
minus the turns of burning of the noun;
    if turns remaining is less than two, say "There's no
portion of [the noun] sufficiently cool for you to pick up."
instead.
```

But that's only a small part of the battle. The thing about fire is that it keeps on doing fiery things even when the player is otherwise occupied: destroying items that are on fire, and spreading to other things nearby. So we need a set of rules for the fire's behavior.

```
Every turn when something is flaming:
    follow the fire rules.


The fire rules is a rulebook.


A fire rule (this is the can't hold flaming objects rule):
    repeat with item running through flaming things:
        if the item is held by the player:
            let turns remaining be the endurance of the item
minus the turns of burning of the item;
            if turns remaining is less than two:
                say "[The item] becomes too hot to hold! ";
                try dropping the item;
```

```
                if the item is held by the player, say "This is
certainly painful."
```

```
A fire rule (this is the flames spread rule):
    repeat with item running through flaming things:
        if the turns of burning of the item is one:
            spread the flames from the item.
```

```
A fire rule (this is the fire destroys things rule):
    now started printing is false;
    repeat with item running through flaming things:
        increment the turns of burning of the item;
        if the turns of burning of the item is greater than the
endurance of the item, destroy the item;
    if started printing is true, say "[paragraph break]";
    now started printing is false.
```

Because we've labelled all the fire rules, we could swap their order, or turn some of them off, while allowing the others run as usual. For instance, if there were a pair of fireproof gloves in the game, we might want to turn off the "can't hold flaming objects rule" whenever the player is wearing them.

This sort of flexibility is especially useful in the context of extensions. Someone writing an extension about burning would have no way of anticipating the need for a Fireproof Gauntlet of Thog, but the author would nonetheless be able to implement one easily.

```
Definition: a thing is vulnerable if it is flammable and it is
whole.
```

The contact between things is a critical factor when it comes to fire, so we might add a couple of conditional relations do determine what is touching what.

```
Reliance relates a thing (called X) to a thing (called Y)
when X is part of Y or X is in Y or X is on Y. The verb to
be relying on means the reliance relation.
```

```
Contact relates a thing (called X) to a thing (called Y)
when X is relying on Y or Y is relying on X. The verb to
be joined to means the contact relation.
```

Having these at our disposal makes it much tidier to write what happens next:

```
To spread the flames from (item - a thing):
    now started printing is false;
    if the item is joined to a flammable whole thing (called
the sacrifice):
        if the sacrifice is visible:
            now started printing is true;
            say "Flames engulf [the list of flammable whole
things which are joined to the item].";
        now all the flammable whole things joined to the
item are flaming.


Started printing is a truth state that varies. Started
printing is false.


To destroy (item - a thing):
    let home be the holder of the item;
    if the item is part of something (called the
superstructure), let home be the holder of the
superstructure;
    if the item is visible:
        now started printing is true;
        say "[The item] burns away[if something is relying
on the item], leaving [a list of things which are relying on
the item] behind[end if]. ";
    if something is relying on the item,
        now all the things which are relying on the item are
in the home;
    now the item is nowhere;
    now the item is damp;
    now every flaming thing which is part of the item is
damp.


To destroy (item - a door):
    let home be the holder of the item;
```

```
    if item is visible:
        now started printing is true;
        say "[The item] burns away[if something flame-
retardant is part of the item], leaving [a list of flame-
retardant parts of the item] behind[end if]. ";
    if home is a room, now all of the flame-retardant parts
of the item are in the home;
    now the item is damp;
    now the item is open;
    now the item is unopenable.



Before printing the name of a damp door:
    say "burnt-out frame of ".



Instead of opening or closing a damp door:
    say "[The noun] can no longer be opened or closed in
any meaningful sense."



Instead of doing something other than examining or
dropping to a flaming thing when the turns of burning of
the noun is greater than 1:
    say "Fire has too thoroughly engulfed [the noun] for
that to be a good idea."



Instead of taking something when the noun is in a
flaming thing (called the receptacle):
    say "You don't quite dare reach into [the receptacle]."



Instead of touching something which is within a flaming
thing (called the receptacle):
    say "It seems a little risky since [the receptacle] is on
fire."



Instead of turning something when the noun is
contained in a flaming thing (called the receptacle):
    say "It seems a little risky since [the receptacle] is on
```

```
fire."


Instead of pushing or pulling something when the noun
is inside a flaming thing (called the receptacle):
    say "[The receptacle] deters you."


Before burning something which is in a container when
the holder of the noun contains the player:
    say "This could make things toasty for you..."
```

And that completes the rules which cover burning: things can catch fire, fire will spread, and gradually consume the world in flames. All of that was general and could be used in any setting, but we now provide a small game to show it off.

```
Part II - Escape from the Library of the Dead


The Library of the Dead is a room. "This room -- little,
dank, stone -- is filling with some miasma you do not
quite dare breathe. It is imperative that you get out."


The desk is a flammable supporter in the Library. A
drawer is part of the desk. The drawer is a flammable
closed container. It is openable, lockable, and locked.
The desk is scenery.


A box is in the Library. A metal hinge is part of the box.
The hinge is flame-retardant. The box is open,
flammable, and openable. The shroud of Laertes is a
flammable thing in the box.


Instead of examining something when something is part
of the noun:
    say "You note [the list of things which are part of the
```

```
noun]."


The world's last manuscript of the Psychagogoi by
Aeschylus is on the desk. The manuscript is flammable.
The manuscript has endurance 1.


The torch is a flammable flaming thing carried by the
player. It has endurance 60. The asbestos sack is a
flame-retardant player's holdall in the drawer.


The trapdoor is up of the Library and east of the Plaza.
The trapdoor is a door. It is flammable, closed, lockable,
and locked. "A trapdoor in the ceiling is your only hope
of escape[if flaming]. Fortunately, it is rapidly burning
through[end if]." The trapdoor has endurance 15.


Instead of going through the closed trapdoor, say "[The
trapdoor] is closed."
```

We can then add a special fire rule to handle the trapdoor, which will be called as part of the same sequence. Again, this would be most important if the fire rules were part of a standard extension, and the trapdoor fire rule the author's own addition.

```
A fire rule:
   if the trapdoor is flaming and a random chance of 1 in
3 succeeds:
      let the caught thing be a random flammable whole
thing which can be touched by the trapdoor;
      if the caught thing is a thing:
         say "A spark from [the trapdoor] catches [the
caught thing]!";
         now the caught thing is flaming.


Instead of going to the Plaza:
   say "Out at last!";
```

end the story finally.

Test me with "get manuscript / get shroud / light desk / look / g / open drawer / look / g / g / g / get sack / put shroud in sack / put manuscript in sack / close sack / light trapdoor / look / g / g / g / g / g / g / g / g / g / g / g / g / g / g / up".

---

402 **ExamplePatient Zero** *RB*

People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

"Patient Zero"

Use the serial comma and no scoring.

Understand "about" as asking for information. Asking for information is an action out of world.

Carry out asking for information: say "An implementation of the following creative brief:

People wander around some small map, on errands. One, sad to tell, has Gelato's Syndrome, a tragic condition turning one's skin the colour of a random flavour of ice cream (raspberry ripple, neapolitan, etc.). When two people are in the same room, there's a 1/3 chance that an infected person will infect a non-infected one. The player can cure any single person: victory condition - to stamp out the disease."

When play begins:
    say "Gelato's Syndrome. It's struck, and it's struck hard. In these sticky summer months, there's no telling who will contract the disease next.";
    now the command prompt is "[if the destination of the player is not blank](heading to [destination of the player]) [end if]>".

Section 1 - Errands

The current actor is a person which varies. The current owner is a person which varies.

Every turn:
    if player is active, follow the character movement rules.

Every turn:
    now the last person named is the player;
    now the last thing named is the player;
    now every person is active.

A person can be active or passive. The player is passive.

The character movement rules are a rulebook.

The first character movement rule:
    now group size is 1;
    now the last person named is the player;
    now the last thing named is the player;
    now the player is passive.

```
A character movement rule:
    repeat with mover running through innocent people:
        now the current actor is the mover;
        follow the shopper rules;
        now the current actor is passive;
    follow the movement reporting rule.


A character movement rule:
    repeat with next mover running through mercantile
people:
        now the current owner is the next mover;
        follow the shopowner rules;
        now the current owner is passive;
    follow the infection rule.


To decide whether movement has not yet occurred:
    if the player is passive, no;
    yes.


Definition: a person is mercantile if it owns a room.
Definition: a person is innocent if it is not mercantile and
it is not the player.


The shopowner rules is a rulebook.


A shopowner rule:
    let the shop be a random room owned by the current
owner;
    if the shop is air-conditioned and an open door (called
the escape) protects the shop, try the current owner
closing the escape instead.


Report someone closing a door when the person asked
owns the location:
```

say "[The person asked], muttering darkly about air-conditioning and electricity, closes [the noun]." instead.

Report Vanessa closing the metal door when the metal door is visible:
if Vanessa is visible, say "Vanessa watches serenely as the metal door slides automatically back in place, sealing Cold Comfort." instead;
otherwise say "The metal door slides heavily back into place." instead.

A shopowner rule:
if the location of the current owner encloses a submitted artwork (called the target):
try the current owner filing the target.

Filing is an action applying to one thing.

Before someone filing something which is not carried by the person asked:
try the person asked taking the noun instead.

Carry out someone filing:
if the person asked does not carry the noun and the person asked is visible, say "[The person asked] tries unsuccessfully to get [the noun]." instead;
now the noun is nowhere.

Report someone filing:
say "[The person asked] registers [the noun] and files it away."

The shopper rules is a rulebook.

A shopper rule:
    if the current actor carries something (called the problem), try the current actor resolving the problem instead.

A shopper rule:
    if the current actor is not in the pool hall and the air conditioner is switched on:
        try the current actor approaching the pool hall;
    otherwise:
        let way be a random direction;
        try the current actor going the way.

Definition: a room is air-conditioned:
    if it is outdoors, no;
    if it is the Pool Hall and the air conditioner is switched off, no;
    if it is protected by a door, yes;
    no.

Protection relates a door (called X) to a room (called Y) when the front side of X is Y or the back side of X is Y. The verb to protect means the protection relation.

Ownership relates one person to various rooms. The verb to own means the ownership relation.

Resolving is an action applying to one thing.

An artwork is a kind of thing. Before printing the name of an artwork, say italic type. After printing the name of an artwork, say roman type. An artwork can be submitted or reserved.

A book is a kind of artwork.

Before someone resolving a book when the person asked is not in the Public Library:
    try the person asked approaching the Public Library instead.

Carry out someone resolving a book:
    move the noun to the Public Library;
    now the noun is submitted.

Report someone resolving a book:
    say "[The person asked] turns in [the noun]."

Before listing contents: group books together.

Before grouping together books: say "books entitled ".

A stamped envelope is a kind of thing.

Before someone resolving a stamped envelope when the person asked is not in the Post Office:
    try the person asked approaching the Post Office instead.

Carry out someone resolving a stamped envelope:
    now the noun is nowhere.

```
Report someone resolving a stamped envelope:
    say "[The person asked] slips [a noun] into the
outgoing mail slot."


Instead of someone resolving a stamped envelope
when the person asked carries at least two stamped
envelopes:
    if the person asked is visible, say "[The person asked]
shoves into the mail slot [a list of stamped envelopes
carried by the person asked].";
    repeat with item running through stamped envelopes
carried by the person asked:
        now the item is nowhere.


A DVD is a kind of artwork.


Before someone resolving a DVD when the person
asked is not in the Rental Store:
    try the person asked approaching the Rental Store
instead.


Carry out someone resolving a DVD:
    now the noun is submitted;
    move the noun to the Movie Rental Store.


Report someone resolving a DVD:
    say "[The person asked] returns [the noun]."


Instead of someone resolving a DVD when the person
asked carries at least two DVDs:
    if the person asked is visible, say "[The person asked]
turns in [a list of DVDs carried by the person asked].";
    now every DVD carried by the person asked is
submitted;
    now every DVD carried by the person asked is in the
location of the person asked.
```

Before listing contents: group DVDs together.

Before grouping together DVDs: say "DVDs of ".

Approaching is an action applying to one thing.

Carry out someone approaching:
    let the way be the best route from the location of the person asked to the noun, using doors;
    if the way is a direction, try the person asked going the way;
    otherwise stop the action.

A coupon is a kind of thing.

Carry out someone resolving a coupon:
    try the person asked giving the noun to Vanessa.

The block giving rule is not listed in any rulebook.

Check giving something to someone (this is the block player giving rule):
    abide by the block giving rule.

Before someone resolving a coupon when the person asked is not in Cold Comfort:
    try the person asked approaching Cold Comfort instead.

```
After someone giving a coupon to Vanessa:
    let the reward be a random ice cream cone;
    let the new flavor be a random infection color;
    now the infection color of the reward is the new flavor;
    move the reward to the person asked;
    now the noun is nowhere;
    if Vanessa is visible, say "[The person asked] trades
in [the noun] and receives [a reward] from Vanessa."


Infection color is a kind of value. The infection colors are
french vanilla, whole-bean vanilla, mint, chocolate, dark
chocolate, chocolate chip, chocolate fudge, mint
chocolate chip, chocolate chocolate chip, triple
chocolate, white chocolate, white chocolate chip, aztec
cocoa-chili, raspberry ripple, neapolitan, rum raisin,
dulce de leche, strawberry chunk, rocky road,
blackberry sorbet, lemon sherbet, lime ice, caramel
swirl, mango, saffron silk, and cookie dough cream.


To say list of flavors:
    let current color be french vanilla;
    while current color is not cookie dough cream:
        say "[current color], ";
        now current color is the infection color after the
current color;
    say "and [current color]".


Understand "ask vanessa for [flavored ice cream]" as
buying the flavor. Understand "buy [flavored ice cream]"
as buying the flavor.


Buying the flavor is an action applying to one infection
color.


Check buying the flavor:
    unless the player can see Vanessa:
        say "It would help if you were in the presence of an
```

ice cream salesperson." instead.

Carry out buying the flavor: say "'Do you have a coupon?' Vanessa demands. You admit you do not. 'No [infection color understood] for you!'"

Understand "ice cream" or "cream" or "ice" or "sherbet" or "sorbet" as "[ice cream]".

Understand "[infection color]" or "[infection color] [ice cream]" as "[flavored ice cream]".

An ice cream cone is a kind of thing. An ice cream cone is always edible. An ice cream cone has an infection color. An ice cream cone can be half-eaten or fresh. Understand the infection color property as referring to an ice cream cone.

Carry out someone resolving an ice cream cone:
    try the person asked eating the noun instead.

Instead of someone eating a fresh ice cream cone:
    now the noun is half-eaten;
    if the person asked is visible, say "[The person asked] licks [the noun]."

Report someone eating an ice cream cone:
    say "[The person asked] pops the end of [the noun] into [if the person asked is female]her[otherwise]his[end if] mouth and swallows." instead.

Before printing the name of an ice cream cone:

say "[if half-eaten]half-eaten [end if][infection color] ".

Section 2 - Infection Rules

This is the infection rule:
    if an infected person (called typhoid mary) can see a clean person (called random bystander) and a random chance of 1 in 3 succeeds:
        try typhoid mary sneezing on the random bystander.

A person can be infected or clean. A person has an infection color.

Every turn:
    if the player is infected, say "You feel itchy."

Definition: a person is other if it is not the player.
Definition: a person is another if it is other.

When play begins: now right hand status line is "Sick: [number of infected people]/[number of people]".

Every turn:
    if every person is infected, end the story saying "Everyone succumbs";
    if every person is clean, end the story finally saying "The Syndrome is eradicated".

Understand "sneeze on [something]" as sneezing on.
Sneezing on is an action applying to one thing.

Check sneezing on:
    if the player is clean, say "You're not sickly." instead;
    if the noun is the player, say "Ew." instead;
    if the noun is not a person, say "[The noun] cannot be infected." instead.

Carry out sneezing on:
    now the noun is infected;
    now the infection color of the noun is a random infection color.

Carry out someone sneezing on:
    now the noun is infected;
    now the infection color of the noun is a random infection color.

Report sneezing on:
    say "Unable to control yourself, you sneeze on [noun].".

Report someone sneezing on:
    say "[The person asked] sneezes on [if the noun is the player]you[otherwise][noun][end if]!".

Understand "inject [someone] with [something]" as injecting it with. Understand "inject [someone] with [syringe]" as injecting it with. Understand "use [syringe] on [someone]" as injecting it with. Understand the commands "innoculate" and "vaccinate" as "inject".

Injecting it with is an action applying to two things.

Check injecting it with:
    if the second noun is not the syringe, say "[The second noun] cannot inject anything." instead;
    if the noun is clean:
        if the noun is the player, say "You're not infected yet." instead;
        say "[The noun] is not infected, and the syringe contains a cure, not a vaccine." instead.

Carry out injecting it with: now the noun is clean.

After injecting the player with something: say "You inject yourself, wincing at the sting. But the itching fades almost at once."

Report injecting it with: say "You inject [the noun], who is now cured (but could easily be reinfected)."

Section 3 - Geography

Include Locksmith by Emily Short.

Understand "go to/toward/into [any room]" as going toward. Understand "enter [any room]" as going toward.

A person has a room called the destination.

Going toward is an action applying to one thing.

```
Check going toward:
    if the noun is the location, say "You're already in [the
location]." instead.

Carry out going toward:
    now the destination of the player is the noun;
    let heading be the best route from the location to the
noun, using even locked doors;
    if heading is not a direction, say "You can't think how
to get there from here." instead;
    try going heading;
    if the location is the destination of the player, now the
destination of the player is blank.

Instead of waiting when the destination of the player is
not blank:
    if the destination of the player is the location:
        now the destination of the player is blank;
    otherwise:
        try going toward destination of the player;
        if the location is the destination of the player, now
the destination of the player is blank.

Understand "stop" or "cease" as stopping. Stopping is
an action applying to nothing. Carry out stopping: now
the destination of the player is blank. Report stopping:
say "You stop in your tracks."

After going to an air-conditioned room:
    say "You step into the mercifully air-conditioned
surroundings of...";
    continue the action.

After going from an air-conditioned room:
    say "You emerge from the air-conditioning into heat
like a wall...";
    continue the action.
```

Instead of listening to an air-conditioned room:
    say "The air-conditioning hums softly."

The Alfred Cralle Pool Hall is a room. "The town's most popular gathering-place, the pool hall is decorated in honor of the inventor of the ice cream scoop." The air conditioner is a device in the Pool Hall. "[if switched off]An air conditioner sits in the corner, unhappily inert[otherwise]The air conditioner hums briskly[end if]."

The felt door is west of the Pool Hall. The felt door is a door. The felt door is open. The felt door is lockable and unlocked. The key to the city unlocks the felt door. The description of the felt door is "It has a prominent lock, designed for an old-fashioned key."

After locking a door with something in the presence of an other person (called audience):
    say "[The audience] looks a little non-plussed when you lock [the noun], but shrugs."

Nancy Johnson Memorial Square is west of the felt door. The description of Nancy Johnson Memorial Square is "Waves of August heat rise from the pavement: more than once you've had the fancy that your shoes are simply going to stick. At the center of the square, rubbed to a brownish polish by many adoring hands, is the statue of Mrs. Nancy Johnson of New Jersey."

The statue is scenery in Memorial Square. Understand "nancy" or "johnson" or "mrs" as the statue. The description of the statue is "Mrs. Johnson is pictured with a hand-cranked ice cream freezer tucked under one arm. Her other hand grips an ice cream scoop, ready to serve frozen dessert to the huddled masses." A hand-cranked ice cream freezer is part of the statue. The description is "The hand-cranked ice cream freezer was Mrs. Johnson's invention in 1846, though it was

William Young who had the sense to patent the thing in 1848." The scoop is part of the statue. The description of the scoop is "An anachronism: Alfred Cralle would not invent the tool until 1897."

The Post Office is northwest of Nancy Johnson Memorial Square. "Service at the post office is on the slow side since everything went automated." The slot is scenery in the post office. The slot is a container. Carry out inserting something into the slot: now the noun is nowhere. Report inserting something into the slot: say "[The noun] falls out of sight, and you know you will never see it again."

Hamwi Street is northeast of an iron gate. "A U-shaped street running from Main Street around to the Memorial Square, Hamwi Street was recently added by ambitious city planners. The small and straggly line of trees has yet to grow enough to provide perceptible shade, so the street is even hotter and more unforgiving than the other parts of town."

The iron gate is northeast of Nancy Johnson Memorial Square. The iron gate is a door. It is lockable and unlocked.

Before printing the name of the iron gate while not opening or closing or locking or unlocking:
    if the person asked is the player:
        if the gate is open, say "open ";
    otherwise if the gate is locked:
        say "locked ";
    otherwise if the gate is closed:
        say "closed ".

Cold Comfort Ice Cream is north of a metal door. The metal door is north of Hamwi Street. A poster is fixed in place in Cold Comfort. "A poster fills one wall with the blazing promise of treats to come." The description of

the poster is "Coming soon! Thai ice creams! Durian, jackfruit, taro, and coconut flavors!"

The metal door is a door. "A frosty metallic door separates [the location] from [the other side of the metal door]." The metal door is lockable and unlocked. The key to the city unlocks the metal door.

Marciony Street is southeast of Nancy Johnson Memorial Square. "A semi-circular terrace, named somewhat fancifully after one claimant to the invention of the ice cream cone -- though Hamwi Street competes for the same honor. There are wedges of cool shadow here and there thanks to the buildings, but for the most part the southern exposure keeps Marciony unpleasantly hot."

The Movie Rental Store is west of a glass door. The glass door is a door. It is west of Marciony Street. The glass door is lockable and unlocked. The key to the city unlocks the glass door.

Main Street is southeast of Hamwi Street. Main Street is northeast of some bronze gates.

The emergency box is in Main Street. The emergency box is fixed in place. "A fire-red box with a glass front faces the sidewalk, with 'In case of emergency, BREAK GLASS' lettered on it." The emergency box is closed and transparent. Understand "glass" as the box. Instead of attacking the closed emergency box: say "You hit the emergency box, which shatters open."; now the emergency box is open. Instead of attacking the open emergency box: say "The glass has already been thoroughly broken."

The syringe is in the emergency box. The description of the syringe is "It contains the cure for Gelato's Syndrome. You can inject anyone you like with it."

The bronze gates are northeast of Marciony Street. The bronze gates are a door. The bronze gates are lockable and unlocked. The description of the bronze gates is "Erected during the milk-taint revolution of 1937, they were designed to keep Main Street safe from the depredations of dairy-starved rioters."

The Public Library is east of Main Street. "Built in the 1920s during the height of the dairy boom, the public library has lush pink velvet seats, marble walls the color of fresh cream, and a motif of cherries carved around every doorframe. An incongruous sign hangs from the ceiling." The incongruous sign is scenery in the Public Library. The description of the incongruous sign is "Eating and drinking in the library is STRICTLY PROHIBITED."

Town Hall is southeast of Main Street. "Town Hall was built during the slow days of the ice-cream bust, and therefore it is as joyless and utilitarian as the Public Library is ridiculous. Unwilling to be reminded of their pain, the inhabitants steered clear of any decoration that might remotely be construed to resemble a scoop of anything: so there are no curves, only disciplined right angles." The key to the city is in Town Hall. It unlocks the iron gate. It unlocks the bronze gates. The description of the key to the city is "A skeleton key."

A room can be indoors or outdoors. The Post Office, the Alfred Cralle Pool Hall, the Store, Cold Comfort, Town Hall, and the Library are indoors.

Use full-length room descriptions.

```
After looking in an outdoors room:
    let started printing be false;
    now every proximate door is not mentioned;
    if an indoors room is adjacent:
        let started printing be true;
        say "From here you can head into [the list of
adjacent indoors rooms][if a proximate door is not
mentioned], or go through [the list of proximate doors
which are not mentioned][end if]. [run paragraph on]";
    if an outdoors room is adjacent:
        say "You could[if started printing is true] also[end if]
go ";
        let count be the number of adjacent outdoors
rooms;
        let index be count;
        repeat with next room running through adjacent
outdoors rooms:
            let way be the best route from the location to the
next room;
            say "[way] to [the next room]";
            decrement index;
        make delimiter index of count, continuing;
    if a proximate door is not mentioned:
        let started printing be true;
        say "[if started printing is true]Also
available[otherwise]Your available exits[end if] [is-are
the list of proximate doors which are not mentioned].";
    otherwise:
        if started printing is true, say paragraph break.



Definition: a door is proximate:
    if the front side of it is the location, yes;
    if the back side of it is the location, yes;
    no.



Before exiting when the player is in an indoors room:
    if the player can see a door (called nearest exit), try
entering the nearest exit instead;
    repeat with way running through directions:
        let next room be the room way from the location;
        if the next room is a room, try going way instead.



Blank is a room. The destination of the player is Blank.
Blank contains 15 ice cream cones.
```

Section 4 - Other Players

Vanessa is a woman in Cold Comfort. Vanessa owns Cold Comfort.

Francine is a woman in the Public Library. Francine carries a book called Phlox for Phyllis. Francine carries a stamped envelope called a pink stamped envelope.

Lewis is a man in the Alfred Cralle Pool Hall. Lewis carries 3 stamped envelopes. Lewis carries a book called Idiot's Guide to Dating. Lewis carries a book called How to Meet Women. Lewis carries a book called Seduction in Three Easy Steps. Lewis carries a DVD called Sleepless in Seattle.

Gene is a man in Nancy Johnson Memorial Square. Gene carries a stamped envelope. Gene carries a DVD called Casablanca. Gene carries a coupon.

Rhoda is a woman in Marciony Street. Rhoda carries a book called The Marciony Street Murders. Rhoda carries a DVD called Unsolved Serial Killings XVIII. Rhoda carries a stamped envelope called a squashy package.

Martin is a man in Main Street. Martin carries a DVD called The Lifecycle of the South Sea Tortoise. Martin carries a coupon.

Antony is a man in Movie Rental. Antony carries a coupon. Antony carries a stamped envelope called a

postcard.

Shelby is a man in the Town Hall. Shelby carries a DVD called Conducting An Orderly Meeting. Shelby carries 5 stamped envelopes. Shelby carries an ice cream cone. Shelby carries a coupon.

Christopher is a man in the Library. Christopher owns the Library.

Linnea is a woman in the Alfred Cralle Pool Hall. Linnea owns the Alfred Cralle Pool Hall.

Ned is a man in the Movie Rental Store. Ned owns the Movie Rental.

After printing the name of someone (called target) while listing contents: if the target owns the location of the target, say " (the owner)".

The description of a person is usually "[The noun] [if the noun is clean]looks healthy[otherwise]is the color of [infection color of the noun][end if]."

After examining another person who is carrying something: say "[if the noun is female]She[otherwise]He[end if] is carrying [a list of things carried by the noun]."

When play begins: let Patient Zero be a random other person; now patient zero is infected.

This is a light variation of a previous example, but we use it here because it is convenient:

```
Section 5 - Conversation

A person has a table name called conversation.

Instead of asking someone about something:
    let the source be the conversation of the noun;
    if topic understood is a topic listed in source:
        if there is a turn stamp entry:
            say "You have already heard that [summary
entry].";
        otherwise:
            now turn stamp entry is the turn count;
            now the character entry is the noun;
            say "[reply entry][paragraph break]";
    otherwise:
        say "[The noun] stares at you blankly.".

Instead of telling someone about something:
    try asking the noun about it.

Understand "recap" or "recall" or "review" as recalling
conversations.

Recalling conversations is an action applying to nothing.

Carry out recalling conversations:
    repeat with speaker running through other people:
        let source be the conversation of the speaker;
        sort source in turn stamp order;
        say "[The speaker] has so far told you: [line
break]";
```

```
	let index be 0;
	repeat through source:
		if there is a turn stamp entry and the speaker is
character entry:
			let index be 1;
			say " [summary entry][line break]";
	if index is 0, say " absolutely nothing[line break]";
	say line break.
```

The conversation of a person is usually Table of General ChitChat.

Table of General ChitChat

[illegible]

[illegible] think we didn't pay our taxes.'"

[illegible] power, har, har.'"

[illegible] replies [the noun]."

[illegible] ter only."

[illegible] funny,' [the noun] replies finally."

[illegible] [the noun]. 'Best ice cream in town, and that's saying something, you bet.'"

[illegible] ws [the noun]. 'Not to everyone's taste, like...' [the noun as pronoun] considers for a moment. 'Like ginger ice cream. Big pieces of crystallized ginger... not everyone likes that.'"

[illegible] since the gummint started making it for us,' says [the noun]. 'Now he just reads off the schedule on the air every morning. Pretty much takes the fun right out of the news, if you ask me.'"

[illegible] don't you?'"

[illegible] I own [the shop],' replies [the noun][otherwise]Work at the creamery, like most folk around here,' answers [the noun]."

[illegible] cookbook section,'" says [the noun]. 'That got censored way back when-- well, way back.'"

The conversation of Vanessa is the Table of Vanessa Chatter.

Table of Vanessa Chatter

[illegible]

[illegible] needing to draw breath."

```
After reading a command:
    while player's command includes "the":
        cut the matched text.
```

This strips 'the' out of the command, so that ASK PERSON ABOUT THE RAIN will be understood as well as ASK PERSON ABOUT RAIN.

Now we try something a bit unusual. Inform on its own will report each action on its own line, so that each character who walks into or out of a room will be described in a separate paragraph. This is usually fine, but in a game with a lot of characters moving around simultaneously, it can become a bit overwhelming. Instead, we may want to condense these reports into a single line, such as "Ben and Jerry enter from the south". The following accomplishes that goal by replacing some of the reporting rules, storing the information in a table, and then reading the table back later, once all the character movement has been resolved and the reports can usefully be collated:

Section 6 - Movement Description

A person has some text called walk style. The walk style of a man is usually "stride". The walk style of a woman is usually "strut". The walk style of Gene is "[one of]wander[or]stroll[purely at random]". The walk style of Francine is "waddle". The walk style of Antony is "scamper". The walk style of Rhoda is "sashay".

Table of Visible Exits

[illegible] chosen

a person [illegible]

with 10 blank rows.

Table of Visible Entrances

[illegible] chosen
[illegible]
with 10 blank rows.

```
To clear (current table - a table name):
    repeat through current table:
        blank out the whole row.

To tidy departures of (current table - a table name):
    let next direction be up;
    repeat through current table:
        if heading chosen entry is next direction:
            let accomplice be character entry;
            choose row with heading chosen of next
direction in the current table;
            if total entry is 1:
                now second entry is accomplice;
                now total entry is 2;
            if total entry is 2:
                unless the second entry is accomplice:
                    now third entry is accomplice;
                    now total entry is 3;
            choose row with character of accomplice in the
current table;
            blank out the whole row;
        otherwise:
            let next direction be heading chosen entry.

A door has a person called last opener.

Report someone opening a door:
    now group size is 1;
    now the last opener of the noun is the person asked;
```

```
    if the person asked is visible, say "[The person asked]
opens [the noun]. [run paragraph on]" instead;
    otherwise say "[The noun] opens from the other side.
[run paragraph on]" instead.


Report someone going through a door (called route):
    if the person asked is not the last opener of the route,
continue the action;
    if the person asked is the last person named, say
"[The person asked as pronoun]";
    otherwise say "[The person asked]";
    say " [if the person asked is in the
location]comes[otherwise]goes[end if] through[if the last
thing named is not the route] [the route][end if]." instead.


The last thing named is a thing that varies. Before
printing the name of something (called target) which is
not a person: now the last thing named is the target.


Report someone going a direction:
    if the person asked is in the location,
        choose a blank row in the table of visible
entrances;
    otherwise choose a blank row in the table of visible
exits;
    now character entry is the person asked;
    now total entry is 1;
    if the person asked is in the location,
        now heading chosen entry is the opposite of the
noun;
    otherwise now heading chosen entry is the noun;
    stop the action.


This is the movement reporting rule:
    sort the Table of Visible Entrances in heading chosen
order;
    tidy departures of the table of visible entrances;
    sort the Table of Visible exits in heading chosen
order;
    tidy departures of the table of visible exits;
    let total row count be the number of filled rows in the
```

```
Table of Visible Entrances plus the number of filled rows
in the Table of Visible Exits;
    if total row count is 0, rule succeeds;
    generate descriptions from the Table of Visible
Entrances;
    generate descriptions from the Table of Visible Exits;
    clear the Table of Visible Entrances; clear the Table
of Visible Exits.



To generate descriptions from (current table - a table
name):
    let count be the number of filled rows in the current
table;
    if count is 0, rule succeeds;
    let index be count;
    repeat through the current table:
        let accomplice be character entry;
        if character entry is a person, now character entry
is marked for listing;
        if there is a second entry and second entry is a
person, now second entry is marked for listing;
        if there is a third entry and third entry is a person,
now third entry is marked for listing;
        let target be the room the heading chosen entry
from the location;
        if total entry is 3, say "[The character entry], [the
second entry][optional comma] and [the third entry] ";
        if total entry is 2, say "[The character entry] and
[the second entry] ";
        if total entry is 1:
            if the character entry is the last person named,
say "[The character entry as pronoun] ";
            otherwise say "[The character entry] ";
        if total entry is 1, say "[walk style of the character
entry]s ";
        otherwise say "walk[if total entry is 1]s[end if] ";
        if the character entry is in the location:
            if location is indoors and target is indoors, say
"over from ";
            if location is outdoors and target is indoors, say
"out of ";
            if location is indoors and target is outdoors, say
"in from ";
            if location is outdoors and target is outdoors, say
"over from ";
        otherwise:
            if location is indoors and target is indoors, say
"over to ";
            if location is outdoors and target is indoors, say
```

```
"into ";
            if location is indoors and target is outdoors, say
"out [if a door is visible][the random visible door][end if]
to ";
            if location is outdoors and target is outdoors, say
"over to ";
        if target is outdoors, say "[the heading chosen
entry]";
        otherwise say "[the target]";
        if the total entry is 1 and count is 1 and accomplice
carries something, say ", carrying [a list of things carried
by the accomplice]";
        decrement index;
        make delimiter index of count, continuing;
        now group size is total entry;
    if a marked for listing person is infected:
        [eliminate the case in which we have already seen
this description because we just typed LOOK and the
patient was in the room at the time]
        if looking and a marked for listing person is not in
the location:
            clear marked people;
            say paragraph break;
        otherwise:
            describe patients;
    otherwise:
        clear marked people;
        say paragraph break.



The last person named is a person that varies. Before
printing the name of a person (called target): now the
last person named is the target. Group size is a number
that varies. Group size is 1.



To clear marked people:
    repeat with named party running through people:
        now the named party is not marked for listing.



Before listing nondescript items:
    if the number of people who are marked for listing is
0, make no decision;
    say "You can see [a list of people who are marked for
listing] here. ";
    now group size is the number of people who are
```

```
marked for listing;
    describe patients;
    now every marked for listing person is not marked for
listing.



To describe patients:
    if every marked for listing person is infected and at
least three people are marked for listing:
        say "They are all sick as dogs, every one.";
        clear marked people;
        rule succeeds;
    otherwise:
        if the number of people who are marked for listing
is greater than two and the number of infected people
who are marked for listing is greater than the number of
clean people who are marked for listing:
            say "Only [the list of clean people who are
marked for listing] currently remain[if the number of
clean people who are marked for listing is 1]s[end if]
untainted.";
            clear marked people;
            rule succeeds;
    let count be the number of marked for listing other
people who are infected;
    if count is 0:
        say paragraph break;
        make no decision;
    let index be count;
    repeat with patient running through marked for listing
other people who are infected:
        if index is count:
            if count is 1 and the patient is the last person
named:
                say "[The patient as pronoun]";
            otherwise:
                say "[The patient]";
        otherwise:
            say "[the patient]";
        say " [looks as though dipped in for index]
[infection color of the patient]";
        decrement index;
        make delimiter index of count;
    clear marked people.



To say (named character - a man) as pronoun:
    if group size is 1, say "He"; if group size is 2, say "The
```

```
latter"; if group size is greater than 2, say "The last".


To say (named character - a woman) as pronoun: if
group size is 1, say "She"; if group size is 2, say "The
latter"; if group size is greater than 2, say "The last".


To say looks as though dipped in for (index - a number):
    let divider be the number of filled rows in the Table of
Dipping Phrases;
    if index is greater than 4, let index be the remainder
after dividing index by divider;
    choose row index in the Table of Dipping Phrases;
    say dipping entry.


Table of Dipping Phrases
dipping
"looks as though dipped in"
"could have been rolling in"
"has a bad case of"
"suffers from"
"contracted a virulent"


A door is usually scenery.
```

The next part could be simpler, but for rigor we will write it in such a way that it will work whether or not the serial comma is in use. This requires some extra work.

```
To make delimiter (index - a number) of (count - a
number), continuing or halting:
    if index is 0:
        if continuing, say ". [run paragraph on]";
        otherwise say ".";
    otherwise if index is 1:
        if count is 2, say " and ";
        otherwise say "[optional comma] and ";
    otherwise:
        say ", ".
```

```
To say optional comma:
    if the serial comma option is active:
        say ",".
```

```
Test me with "go to cold comfort / z / z / z / z / ask
vanessa for french vanilla / ask vanessa for chocolate /
ask vanessa about flavors / ask vanessa for chocolate
chocolate chip".
```

Because so much of this game is randomized, it will not be possible to provide a test command that systematically solves it. A good strategy is to go to Main Street, get the syringe; go to the Town Hall and get the key; then visit the shops, inject everyone, and lock them in when they've all been injected. Then go to the Pool Hall, turn on the air conditioner, and wait for the remaining parties to show up.

This is also something that could get fairly slow if we added many more rooms and characters to it. In that case, we might want to select fast route-finding so that character movement won't take as long. This will cost memory, possibly forcing the game into Glulx format if it isn't already, but significantly reduce the run-time for large maps with numerous people moving each turn:

```
Use fast route-finding.
```

ExampleFlotation

RB

Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

Here we want a rulebook to determine whether objects float or sink, so we create an object-based rulebook for the purpose. The more specific rules here, pertaining to corks and to inflated things, will be consulted first; then, as a default, the general flotation rule.

We also want a switch that can turn flotation off at will. The rule about the big switch will be observed before the others because the when... clause makes it more specific than the other rules in the flotation rulebook.

If we wanted, we could also put these rules into a rulebook in an explicit order, overriding Inform's automatic sorting by specificity.

```
"Flotation"

The Pumping House is a room.

A well is a fixed in place container in the Pumping
House.

Instead of examining the well:
    say "[if something is in the well]On the surface of the
water you can see [a list of things in the well]
[otherwise]There is nothing on the surface of the water,
nor can you see into the depths[end if]."

The well bottom is a container.

The cork, the rubber ring and a lead ingot are in the
Pumping House.

A big switch is a fixed in place device in the Pumping
House. "A big switch labelled 'MAKE EVERYTHING
SINK' is mounted on one wall[if switched on]. It crackles
with electricity[otherwise]. It is currently switched off and
silent[end if]."

A thing can be inflated or uninflated. A thing is usually
uninflated. Before printing the name of an inflated thing:
say "inflated ".
```

The rubber ring is inflated.

The flotation rules are an object-based rulebook.

A flotation rule for the cork: rule succeeds.
A flotation rule for an inflated thing: rule succeeds.
A flotation rule when the big switch is switched on: rule fails.

After inserting something into the well:
    follow the flotation rules for the noun;
    if the rule succeeded:
        say "[The noun] bobs on the surface.";
    otherwise:
        move the noun to the well bottom;
        say "[The noun] sinks out of sight."

A thing can be sinking, rising, or static. A thing is usually static.

Definition: a thing is wet:
    if it is in the well, yes;
    if it is in the well bottom, yes;
    no.

Every turn:
    now every thing is static;
    repeat with item running through wet things:
        follow the flotation rules for the item;
        if the rule failed and the item is in the well, now the item is sinking;
        if the rule succeeded and the item is in the well bottom, now the item is rising;
    now every rising thing is in the well;

```
    now every sinking thing is in the well bottom;
    if something is rising, say "[The list of rising things]
rise[if the number of rising things is 1]s[end if] to the
surface of the well.";
    if something is sinking, say "[The list of sinking things]
sink[if the number of sinking things is 1]s[end if] out of
sight."
```

And finally a few description rules to make things look prettier:

```
Rule for writing a paragraph about the well when the
well contains something:
    say "The chief feature of the room is a concrete-sided
well in which there float[if the number of things in the
well is 1]s[end if] [a list of things in the well]."


Rule for writing a paragraph about the well:
    say "The chief feature of the room is a concrete-sided
well full of water."
```

As we recall from the chapter on activities, "writing a paragraph about..." is an activity; activities are themselves structured as sets of object-based rulebooks. The activity "writing a paragraph about" uses three object-based rulebooks (before writing..., for writing..., after writing...). We could have made a flotation activity as well, but in general it is overkill to make an activity to make success/failure decisions. For that purpose an object-based rulebook is sufficient.

```
Test me with "get all / put cork in well / put ring in well /
put ingot in well / x well / get cork / get ring / switch
switch on / put cork in well / put ring in well / x well /
switch switch off / switch switch on".
```

ExampleKyoto

RB

Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

Suppose we want to expand the function of the existing THROW

SOMETHING AT command so that a thrown object actually does make contact most of the time. A glance at the Actions index tells us that the Throwing it at rulebook currently looks like this:

```
Throwing something at something (past tense thrown it
at)
    "drop [something held] at/against/on/onto
[something]"

implicitly remove worn clothing rule
futile to throw things at inanimate objects rule
block throwing at rule
```

Some of those still look useful. We want to leave the "implicitly remove thrown clothing" rule, for instance -- no fair having the player throw a hat that's on his head. On the other hand, the "futile to throw things at inanimate objects rule" is going to have to go, because that would prevent us from ever being able to complete the throwing command. So let's get rid of that:

```
"Kyoto"

Part 1 - Throwing Rules

The futile to throw things at inanimate objects rule is not
listed in the check throwing it at rules.
```

That "block throwing at" rule also looks sinister: any "block..." rule in the standard actions library is there to print a message telling the player he can't do what he's asked to do.

But it's not enough to ignore it, the way we did the "futile" rule. Since we are only expanding the command to affect inanimate objects, let's replace the "block throwing at" rule with a different one which will only prevent the player throwing things at people:

```
The block throwing at people rule is listed instead of the
```

```
block throwing at rule in the check throwing it at rules.
```

```
This is the block throwing at people rule:
    if the second noun is a person, say "That might be
construed as an attack." instead.
```

Now we've changed the command so that some action can sometimes be carried out here -- but we don't have any rules for what happens. It's time to create some rules for our model world.

```
A thing can be hard or soft. A thing can be fragile or
strong. Shape is a kind of value. The shapes are round,
flat, and linear. A thing has shape.
```

If we're actually going to allow throwing, we might want to add a couple of extra checks to the rulebook to make sure that this happens when it ought:

```
Check throwing it at (this is the block juggling rule):
    if the player is carrying the second noun, say "It
would be difficult to throw at something you are yourself
holding." instead.
```

```
Check throwing it at (this is the avoid throwing things
into themselves rule):
    if the second noun is within the noun, say "That would
be a nice magic trick." instead.
```

And then the rules for the action itself:

```
Carry out throwing it at (this is the check aerodynamics
rule):
    if the noun is flat:
        move noun to location;
        say "[The noun], unwieldy, flutters to the ground.";
        rule succeeds.
```

That "rule succeeds" ends the action here, if the noun is flat. If not, Inform goes on to the next rule in the carry out throwing it at rulebook:

```
Carry out throwing it at (this is the contact rule):
    say "[The noun] hits [the second noun].[paragraph
break]";
    if the second noun is fragile and the noun is hard:
        destroy the second noun.
```

```
Carry out throwing it at (this is the landing rule):
    let destination be the location;
    if the second noun is on a supporter (called endtable),
let destination be the endtable;
    if the second noun is a supporter, let destination be
the second noun;
    move the noun to the destination;
    if the noun is fragile and the second noun is hard:
        destroy the noun;
        rule succeeds;
    say "[The noun] lands [if the destination is the
location]nearby[otherwise]on [the destination][end if]."
```

These rules are assuming some backup information, so let's provide that as well:

```
Reliance relates a thing (called X) to a thing (called Y)
when X is part of Y or X is in Y or X is on Y. The verb to
be relying on means the reliance relation.
```

```
To destroy (item - a thing):
    let home be the holder of the item;
    if the item is part of something (called the
superstructure), let home be the holder of the
superstructure;
    if the item is visible:
        say "[The item] breaks[if something is relying on
the item], leaving [a list of things which are relying on
the item] behind[end if].";
    if something is relying on the item,
        now all the things which are relying on the item are
in the home;
    now the item is nowhere.
```

Now suppose we'd like to add some further cases for what happens if the

player breaks a fragile door this way:

```
To destroy (item - a door):
    now the item is open;
    now the item is unopenable;
    say "[The item] smashes."


Rule for printing the name of an unopenable open door
while not throwing something at something:
    say "open doorway".


Understand "door" or "doorway" as a door.
```

This works, except that objects will continue to "strike" open, unopenable doors, with the result that the player can smash the same door over and over. What we need is another rule, after the aerodynamics rule and before the contact rule, that tells Inform how to handle throwing things at open doors.

```
This is the flying through doorways rule:
    if the second noun is an open door:
        let the distant room be the other side of the second
noun;
        move the noun to the distant room;
        say "[The noun] flies out of sight into [the distant
room].";
        rule succeeds.
```

If the original rulebook is one we wrote ourselves, we could just add that rule in the proper spot in order. If we got it from an extension, though, we might need to put it in the right place explicitly:

```
The flying through doorways rule is listed before the
contact rule in the carry out throwing it at rules.
```

The magic of rulebooks is that they allow authors to amend each other's work (or the Standard Rules) with a fair amount of freedom. A well-written extension will give individual names to its rules, to allow subsequent authors to modify the function of the extension without too

much trouble.

Now for an actual scenario with which to test this:

Part 2 - The Study

The sliding paper screen is a door. It is north of the Moss Garden and south of the Study. The paper screen is fragile.

The player carries a netsuke and a shamisen. The description of the netsuke is "A weight for the cord on which you wear your purse or your medicine box. This particular one has the shape of a bullfrog, carved from green stone." The netsuke is round, hard, and strong. Understand "green" or "stone" or "bullfrog" as the netsuke.

The description of the shamisen is "An instrument you have only begun to learn to play." The shamisen is linear, soft, and fragile. A neck is part of the shamisen. The neck is linear, strong, and hard. A body is part of the shamisen. The body is round, fragile, and soft. A string is part of the shamisen. The string is linear, soft, and strong. The printed name of the body is "[if the body is not part of the shamisen]shamisen [end if]body". The printed name of the neck is "[if the neck is not part of the shamisen]shamisen [end if]neck". Understand "shamisen" as the body when the body is not part of the shamisen. Understand "shamisen" as the neck when the neck is not part of the shamisen.

The description of the Study is "A restful three-tatami room." The Study contains a calligraphy box and a hanging scroll. The initial appearance of the hanging scroll is "A handsome scroll depicts two women in kimonos crossing a bridge; Mount Fuji is in the background." The calligraphy box contains a brush. The box is openable and closed. The brush is hard, linear,

and strong. The calligraphy box is round, soft, and strong. The hanging scroll is flat, soft, and strong.

The description of the Moss Garden is "Earlier today, you arranged three leaves on the moss in imitation of autumn. They must not be disturbed." The leaves are scenery in the Moss Garden. Instead of throwing something at the leaves: say "You spent too long over their placement."

Test me with "test one / test two".

Test one with "open screen / throw netsuke at screen / n / get netsuke / close screen / get scroll / throw scroll at screen / throw netsuke at scroll / get netsuke / throw netsuke at shamisen / drop netsuke".

Test two with "throw shamisen at netsuke / get all / throw netsuke at screen / get netsuke / throw netsuke at door / s / get netsuke".

---

405 **Example** **Being Peter** *RB*

A set of rules determining the attitude a character will take when asked about certain topics.

Let's say that we're implementing a particularly irrational and volatile character. Some of the time she remains composed; some of the time she reacts with unexpected vehemence for reasons only partly related to what was said.

Moreover, her responses are divided between successful and failing outcomes, where success indicates that she's not too upset and failure means that she is distraught; we use this to determine how the rest of the room reacts.

```
"Being Peter"


The Drawing Room is a room. "The company is
assembled here for champagne. Most of it, anyway:
Mary is on the phone to her babysitter, Roger is keeping
her anxious company, and Carol doesn't drink. But
everyone else."


Maggie is a woman in the Drawing Room.


The player wears a top hat.


Quizzing it about is an action applying to one thing and
one visible thing. Understand "ask [someone] about
[any thing]" as quizzing it about.


Instead of quizzing Maggie about something:
    follow the attitude rules;
    say "Everyone waits to see what the reaction will be:
[outcome of the rulebook].";
    if rule succeeded, say "There is general relief.";
    otherwise say "Everyone is pointedly silent."


The attitude rules are a rulebook. The attitude rules
have outcomes she stays calm (no outcome - default),
she gets angry (failure), she has a stroke (failure), she is
only mildly annoyed (success), and she is elated
(success).
```

Here we want Inform to consult every appropriate attitude rule until it gets to some answer; if an attitude rule does not provide a result, the default 'no outcome' will mean that we go on to the next rule, and so on.

```
A subject is a kind of thing. income, love life, and
```

```
children are subjects.
```

```
An attitude rule for quizzing Maggie about love life:
    she gets angry.
```

```
An attitude rule:
    if the player wears the top hat, she gets angry.
```

Now, as we saw, the 'no outcome' result will never be returned and printed as Maggie's reaction, precisely because it is "no outcome". Therefore, we provide a final attitude rule which will give her a default response to all statements:

```
The last attitude rule:
    she is only mildly annoyed.
```

```
Test me with "ask maggie about love / ask maggie
about income / take off hat / ask maggie about income".
```

There are plenty of contexts where we might want named outcomes for clarity but not want to print the results literally afterward.

Example **Feline Behavior** *RB*

A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

Suppose we have a cat which is supposed to react to (and destroy) the most interesting thing in its environment. There are several ways we could approach this problem, but for the sake of demonstration, let's have it follow a rulebook to figure out which thing it most wants to interact with. We will then return the chosen object as "the object produced by the cat behavior rules".

```
"Feline Behavior"
```

The Kitchen is a room. The cat is an animal in the Kitchen. In the Kitchen is a bowl, a ball of wool, a newspaper. The bowl contains a quantity of cream.

The cat is wearing a silver collar. The description of the cat is "It is wearing [a list of things worn by the cat]."

The player carries a closed openable container called a bag. The bag contains catnip.

The cat behavior rules is a rulebook producing an object.

A cat behavior rule when the cat can touch the catnip:
    say "The cat frolics with the catnip until nothing remains of it.";
    rule succeeds with result catnip.

A cat behavior rule when the cat can touch the cream:
    say "The cat laps up the cream.";
    rule succeeds with result cream.

A cat behavior rule when the cat can touch the ball of wool:
    say "The cat makes the ball of wool into a useless tangle which must be discarded.";
    rule succeeds with result ball.

A cat behavior rule when the cat can touch the newspaper:
    say "The cat bats playfully at the newspaper until all the nasty boring articles are destroyed.";

```
    rule succeeds with result newspaper.


A cat behavior rule:
    say "The cat looks miffed at the lack of ready
entertainment, and glares at you with yellow eyes as
though wondering whether your pants leg is good for
claw-sharpening.";
    rule fails.


Every turn:
    let the destroyed object be the object produced by the
cat behavior rules;
    if the destroyed object is not nothing:
        now the destroyed object is nowhere;
        say "[line break]Good thing you have no use for
[the destroyed object] yourself.[paragraph break]".


Test me with "z / z / open bag / z / z".
```

We include the if rule succeeded... condition here because nothing will be returned if the cat's search failed (as for instance in the result of the final rule).

Naturally, if we wanted we could equally well ask "if rule failed...".

---

407 **ExampleTilt 2** *RB*

A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

In our previous implementations of playing cards, we've gotten as far as creating decks of individual cards that the player can draw and discard. But in a poker game, one doesn't just have a collection of cards: one has a hand of a specific kind.

Here we take on the job of writing an inventory listing for a poker hand that will reflect the real value of what the player has drawn. To do this, we create a rulebook to sort and assess the cards in the player's hand; its possible return values are limited to the kinds of poker hands that exist,

from "high card" to "royal flush".

The first three sections, creating the deck of cards and the means to parse their names, are identical to those we've already seen in Tilt 1; new material begins at section 4.

For the purposes of demonstration, we're simulating something akin to five-card draw without wilds; stud or hold-em variations would add some other complexities.

```
"Tilt"


Section 1 - Cards


Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades. Understand "heart" as hearts.
Understand "club" as clubs. Understand "diamond" as
diamonds. Understand "spade" as spades.


A card is a kind of thing. A card has a suit. A card has a
number called rank. Understand the suit property as
describing a card. Understand the rank property as
describing a card.


52 cards are in the card repository.


To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".


Rule for printing the name of a card (called target):
    say "[rank of the target as a card value] of [suit of the
target]"
```

Table of Value Names

topic	value
"1"	"ace/A/one"
"2"	"deuce/two"
"3"	"three"
"4"	"four"
"5"	"five"
"6"	"six"
"7"	"seven"
"8"	"eight"
"9"	"nine"
"10"	"ten"
"jack"	"knave/J"
"queen"	"Q"
"king"	"K"

After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
            cut the matched text;
    repeat through the Table of Value Names:
        while the player's command includes topic entry:
            replace the matched text with value entry.

When play begins:
    reconstitute deck.

To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;
            now item is in the deck of cards;
        now current suit is the suit after the current suit.

Section 2 - The Deck and the Discard Pile

The Empty Room is a room. "Nothing to see here."

The deck of cards is in the Empty Room. It is a closed unopenable container. The description is "A standard poker deck."

The discard pile is a closed unopenable container. The description is "Cards in this game are discarded face-down, so the discard pile is not very interesting to see. All you can observe is that it currently contains [if the number of cards which are in the discard pile is less than ten][the number of cards which are in the discard pile in words][otherwise]about [the rounded number of cards which are in the discard pile in words][end if] card[s]."

To decide what number is the rounded number of (described set - a description of objects):
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.

Rule for printing room description details of something: do nothing instead.

Section 3 - Drawing and Discarding Actions

Understand the commands "take" and "carry" and "hold" and "get" and "drop" and "throw" and "discard" as something new.

Understand "take [text]" or "get [text]" or "drop [text]" as a mistake ("Here, you only draw and discard. Nothing else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as drawing. Drawing is an action applying to nothing. The drawing action has an object called the card drawn.

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

Understand "discard [card]" as discarding. Discarding is an action applying to one thing.

```
Check discarding:
    if the player does not carry the noun, say "You can
only discard cards from your own hand." instead.


Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile
to the location.


Report discarding:
    say "You toss [the noun] nonchalantly onto the
discard pile."
```

New material begins here. We want to start by grouping cards together, but identifying poker hands only if the player holds a full five cards.

```
Section 4 - Assessing Hands


Before listing contents while taking inventory: group
cards together.


Before grouping together cards:
    if the number of cards carried by the player is 5:
        say "[run paragraph on]";
        follow the hand-ranking rules;
        if the rule succeeded, say "[the outcome of the
rulebook]";
        otherwise say "some random cards";
        if the outcome of the rulebook is pair outcome, say
" of [rank of the first thing held by the player as a card
value]s";
    otherwise:
        say "[number of cards carried by the player in
words] assorted cards";
    say " (".
```

```
Rule for grouping together cards:
    say "[list hand]".


To say list hand:
    let chosen card be the first thing held by the player;
    while chosen card is a card:
        say "[chosen card]";
        now chosen card is the next thing held after
chosen card;
        if chosen card is a card, say ", ".


After grouping together cards:
    say ")".
```

The ranking of poker hands traditionally depends on three features: 1) whether all the cards are of the same suit (flush); 2) whether the cards constitute a numerical run of ranks (straight); and 3) how many cards or sets of cards are of matching rank (pairs, three of a kind, and four of a kind). Here we will start by assessing our hand to determine these qualities:

```
The hand-ranking rules is a rulebook. The hand-ranking
rules have outcomes royal flush, straight flush, four of a
kind, full house, flush, straight, three of a kind, two pairs,
pair, high card.


The hand-ranking rulebook has a truth state called the
flushness.
The hand-ranking rulebook has a truth state called the
straightness.


The hand-ranking rulebook has a number called the pair
count.
The hand-ranking rulebook has a number called the
triple count.
The hand-ranking rulebook has a number called the
```

```
quadruple count.
```

For convenience in identifying hand features, and for elegance when we print the hand-listing, we start by sorting the cards in the player's hand so that the high-ranked cards are listed first. It is rare that we want to concern ourselves with this, but as we saw in the section on "Looking at containment by hand" in the chapter on Change, Inform keeps an ordered list of the items inside any given container; so it does order the objects in the player's hand, and the ordering depends on which things were added to the hand most recently. By moving something to the player's hand again (even if it was already there), we change this ordering, and wind up with a sorted hand.

```
A card can be sorted or unsorted. A card is usually
unsorted.
```

```
Definition: a card is high if its rank is 11 or more.
Definition: a card is low if its rank is 4 or less.
```

```
A hand-ranking rule (this is the initial sort rule):
    now every card is unsorted;
    while the player carries an unsorted card:
        let item be the lowest unsorted card held by the
player;
        move item to the player;
        now the item is sorted;
    if sort-debugging is true, say "-- after initial sort: [list
hand]".
```

This last printing instruction is there for diagnostic purposes: later we'll add a testing command to turn debugging on and off; when it's on, the game will print out its card list at various stages in sorting, to help us trouble-shoot any problems. In normal play, however, this will be off.

Next up, a check to see whether the player has a flush:

```
A hand-ranking rule (this is the finding flushness rule):
    let called suit be the suit of a random card carried by
the player;
    if every card carried by the player is called suit, now
flushness is true.
```

Now we check for straights; this is slightly complicated by the fact that an ace can be either the bottom of a low straight (lower than 2) or the top of a high straight (higher than king), so we explicitly check both possibilities.

```
A hand-ranking rule (this is the finding straightness
rule):
    now straightness is true;
    let N be the rank of the highest card which is carried
by the player;
    repeat with current rank running from N - 4 to N:
        now the test rank is the current rank;
        unless the player carries a matching card:
            if the current rank is N - 4 and the current rank is
9 and the player carries an ace card, do nothing; [this
covers the case where an ace could be the top card of
the sequence]
            otherwise now straightness is false.
```

And finally, we need to identify any groups of cards of the same rank. We want to know how many groups there are and how large each group is (though in practice there can only be one group of three or four in a standard-sized poker hand). We also want to mark any grouped cards so that we can move them to the front of the player's hand when we take inventory.

```
A card can be quadrupled, tripled, paired or
uncombined.
```

```
Test rank is a number that varies. Definition: a card is
matching if its rank is the test rank.
```

This definition is a convenience so that we don't have to write so many explicit loops in the following rule:

```
A hand-ranking rule (this is the counting multiples rule):
    now every card is uncombined;
    repeat with current rank running from 1 to 13:
        now test rank is current rank;
        let N be the number of matching cards held by the
player;
        if N is 4:
```

```
            increment the quadruple count;
            now every matching card held by the player is
quadrupled;
        if N is 3:
            increment the triple count;
            now every matching card held by the player is
tripled;
        if N is 2:
            increment the pair count;
            now every matching card held by the player is
paired.
```

Next we tweak our sorting to reflect the make-up of the hand. There are two reasons why this might differ from the straight highest-to-lowest sort we did earlier:

1) we want to list aces as high unless they are serving as the bottom of a low straight, in which case they should appear last;

2) we want combinations to appear at the front of the list, sorted from highest value to lowest value: larger combinations first, then smaller combinations, and combinations of equal size sorted by rank.

```
A hand-ranking rule (this is the move aces up unless
there's a low straight rule):
    unless the straightness is true and the lowest card
carried by the player is an ace card and the rank of the
highest card carried by the player is 5,
        now every ace card which is carried by the player
is carried by the player;
    if sort-debugging is true, say "-- after ace movement
rule: [list hand]".
```

```
A hand-ranking rule (this is the move pairs forward rule):
    while the player carries a paired card:
        let selection be the lowest paired card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after pairs movement:
[list hand]".
```

```
A hand-ranking rule (this is the raise ace pairs rule):
    if the player carries exactly two ace cards:
```

```
        repeat with item running through ace cards which
    are carried by the player:
            move item to the player;
    if sort-debugging is true, say "-- after paired-ace
    movement: [list hand]".


    A hand-ranking rule (this is the move multiples forward
    rule):
    while the player carries a tripled card:
        let selection be the lowest tripled card which is
    carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    while the player carries a quadrupled card:
        let selection be the lowest quadrupled card which
    is carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after multiples
    movement rule: [list hand]".


    Definition: a card is ace if its rank is 1.
    Definition: a card is king if its rank is 13.
```

Now, having determined the salient qualities of our hand, we run through rules in order from the highest kind of poker combination to the lowest. Because of the order of the source, Inform will choose whichever combination applies first.

```
    A hand-ranking rule (this is the royal-flush rule):
    if flushness is true and straightness is true and the
    highest card carried by the player is king and the lowest
    card carried by the player is ace, royal flush.


    A hand-ranking rule (this is the straight-flushes rule):
    if flushness is true and straightness is true, straight
    flush.


    A hand-ranking rule (this is the four-of-a-kind rule):
```

```
    if the quadruple count is 1, four of a kind.

A hand-ranking rule (this is the full-house rule):
    if the pair count is 1 and the triple count is 1, full
house.

A hand-ranking rule (this is the flushes rule):
    if flushness is true, flush.

A hand-ranking rule (this is the straights rule):
    if straightness is true, straight.

A hand-ranking rule (this is the three-of-a-kind rule):
    if triple count is 1, three of a kind.

A hand-ranking rule (this is the two-pair rule):
    if the pair count is 2, two pairs.

A hand-ranking rule (this is the pair rule):
    if the pair count is 1, pair.

A hand-ranking rule (this is the default rule):
    high card.
```

And finally, we need to define our debugging variable here, even though we won't give the player the ability to turn it on and off except in the special testing section.

```
Sort-debugging is a truth state that varies.
```

For many examples, a test-me script is enough to prove that the example

does what it ought. This example, though, is a bit more complicated, and hard to test randomly. The remainder of the source here shows how we might write a test to verify the desired behavior of our rulebook. Those who are only interested in the rulebook itself can stop reading at this point.

Section 5 - Testing hand identification - Not for release

For the sake of testing our rules, we provide an apparatus that will load the player's hand up with sample hands of each kind, then show the result to make sure that the hand is being correctly identified.

```
Understand "debug sorting" as debugging hand sorting.
Debugging hand sorting is an action out of world.

Carry out debugging hand sorting:
    if sort-debugging is false, now sort-debugging is true;
    otherwise now sort-debugging is false.

Report debugging hand sorting:
    say "Sort debugging is now [if sort-debugging is
true]on[otherwise]off[end if]."

Test me with "draw / g / g / g / g / force hand / g / g / g /
g / g / g / g / g / g / g / g / g".
```

The somewhat rough-and-ready principle of this table is that we will overwrite the cards in the player's hand by resetting their ranks and suits; every five rows of the table represent a new poker hand for the game to attempt to sort and identify. These include one example of each of the major kinds of poker hand, plus a couple of variations involving aces which test the special sorting rules.

```
Table of Testing Hands

set rank  set suit
```

[royal flush]
13 spades
12 spades
11 spades
10 spades
[straight flush]
12 clubs
11 clubs
10 clubs
9 clubs
8 clubs
[four of a kind]
5 hearts
5 spades
5 clubs
5 clubs
[full house]
1 spades
1 hearts
10 spades
10 clubs
[flush]
5 hearts
7 hearts
11 hearts
12 hearts
[straight]
13 spades
12 diamonds
11 clubs
10 hearts
[three of a kind]
2 spades
2 clubs
4 clubs
3 spades
[two pair]
6 spades
7 clubs
7 diamonds
9 hearts
[two pair, ace high]
6 spades
1 clubs
7 diamonds
1 hearts
[pair]
12 spades
12 diamonds
3 spades
4 clubs
[high]
13 diamonds

14 hearts
9 spades
7 clubs
5 diamonds
[testing sorting: low straight]
2 diamonds
3 spades
4 diamonds
5 diamonds

Understand "force hand" as forcing a hand. Forcing a hand is an action out of world.

Current marker is a number that varies.

Carry out forcing a hand:
    repeat with item running through cards which are carried by the player:
        increment current marker;
        if current marker is greater than the number of filled rows in the Table of Testing Hands, now current marker is 1;
        choose row current marker in the Table of Testing Hands;
        now the suit of item is the set suit entry;
        now the rank of item is the set rank entry.

Report forcing a hand:
    try taking inventory.

ExampleElectrified

Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances. *RB*

Suppose we want to prevent the player from touching anything electrified

-- not just as a response to TOUCH OBJECT, but at any time when the action would require contact with the object in question.

"Electrified"

A thing can be safe or electrified. A thing is usually safe.

The Open Field is a room. "At this end of the field is a wire fence separating farm country from the government testing grounds beyond." The wire fence is an electrified thing in Open Field. It is scenery. The description of the wire fence is "Built into the fence is [a list of things which are part of the fence]." The scary box is an electrified container. It is part of wire fence. In the scary box is an alluring prize.

The player carries a flashlight, a grappling hook, a very thick rubber glove, and a length of rope. The glove is wearable.

This is the electrocution-wisdom rule:
    if the player wears the very thick rubber glove, make no decision;
    if the action requires a touchable noun and the noun is electrified, say "You fear touching [the noun]." instead;
    if the action requires a touchable second noun and the second noun is electrified, say "You fear touching [the second noun]." instead.

The electrocution-wisdom rule is listed before the basic accessibility rule in the action-processing rules.

Before touching the scary box:

```
    say "You can't help noticing a bright red sticker on the
surface of the box." [This rule will fire even if we are not
wearing the glove, because Before rules occur before
basic accessibility.]

Instead of opening the scary box:
    say "The scary box seems to be super-glued shut."
[This one won't, because Instead rules occur after basic
accessibility.]

Test me with "touch fence / touch box / open box / wear
glove / open box".
```

---

409 **ExampleTimeless** *RB*

A set of actions which do not take any game time at all.

In a game with tight timing, it is sometimes friendliest to the player to let him LOOK and EXAMINE as much as necessary without being penalized.

```
"Timeless"

Examining something is acting fast. Looking is acting
fast.
```

Now we need a rule which, just at the right moment, stops the turn sequence rulebook in the cast of our new fast-acting actions:

```
The take visual actions out of world rule is listed before
the every turn stage rule in the turn sequence rules.
```

```
This is the take visual actions out of world rule: if acting
fast, rule succeeds.
```

Thus the rest of the turn sequence rulebook is omitted for looking or examining: in effect, they become out-of-world actions like "saving the game". If we wanted to add, say, taking inventory to the list of instant activities, we would just need to define it as acting fast, too.

Now the scenario for testing:

```
When play begins:
    say "You are cornered by a pack of zombie wolves,
armed only with a torch and a pair of pinking shears.
This may be your last moment on earth, unless you can
think fast!"


Cleft is a room. "You're backed into a cleft in the granite:
behind you are only steep, high faces of stone, and
before you a narrow passage."


The plural of zombie wolf is zombie wolves. A zombie
wolf is a kind of animal. Four zombie wolves are in Cleft.


Rule for writing a paragraph about zombie wolves:
    say "The good news is that there isn't much space in
which for the zombie wolves to attack.";
    now every zombie wolf is mentioned.


A steep high face of stone is scenery in Cleft.
Understand "rock" as the stone. The description is "Now
that you look more closely, there appear to be pitons
driven into the rock."


Some pitons are part of the stone. The description of the
pitons is "It looks as though someone else has made
this ascent before."
```

```
Instead of climbing the stone, try going up. Instead of
climbing the pitons, try going up.


Above the Cleft is Clifftop.


Every turn when the location is Cleft:
    say "Alas, your time has run out. The alpha wolf
springs--";
    end the story.


Every turn when the location is Clifftop:
    say "After a breathless climb, you emerge at last onto
the open clifftop.";
    end the story finally.


Test me with "x me / x stone / x pitons / climb pitons".
```

---

410 ExampleEndurance

RB

Giving different actions a range of durations using a time allotment rulebook.

Here we move to a systematic way of giving different durations to different actions, including even variations on the same act -- so that for instance climbing a steep hill might take several minutes more than other going actions. We do this by setting a number, "work duration", to represent the number of minutes consumed by a given action, and then consulting a rulebook to find out how long the past turn's action should take. By default, an action will take 1 minute.

We'll start by emulating the behavior of "Uptempo": each turn we'll set the clock forward most of the way, then check to see what has changed since the last turn, print any relevant events, and only then set the clock forward the final minute. The exception is when an action is set to take no time at all; in that case, we'll skip the rest of the turn sequence rules

entirely.

"Endurance"

Work duration is a number that varies.

Every turn:
    now work duration is 0;
    increment the turn count;
    follow the time allotment rules;
    if work duration is 0, rule succeeds;
    increase the time of day by (work duration minutes - 1 minute).

The time allotment rules are a rulebook.

A time allotment rule for examining or looking:
    now work duration is 0;
    rule succeeds.

A time allotment rule for going:
    now work duration is 2;
    rule succeeds.

A time allotment rule for going up:
    now work duration is 5;
    rule succeeds.

A time allotment rule for waiting:
    now work duration is 10;
    rule succeeds.

The last time allotment rule:
    now work duration is 1.

When play begins: now the right hand status line is "[time of day]".

The Quai is a room. "An attractive park at the edge of the river Aude: here you can wander among palm trees, and watch cyclists go by on the bike path; in the water there are ducks. In the cafe to your north, patrons sip their pastis; and above you is the medieval walled city and its castle."

The Cafe is north of the Quai. "A charming collection of umbrella-shaded tables, from which one can watch the river and the walls of the city beyond. The noise of traffic is only a minor distraction."

The City is above the Quai.

After going to the City:
    say "You struggle uphill for some distance...";
    continue the action.

At 9:15 AM:
    say "The bells ring out from Place Carnot."

Test me with "z / n / s / u".

Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

```
"Escape from the Seraglio"


Section 1 - Special Announcement Rules


The number of takes this turn is a number that varies.
Every turn: now the number of takes this turn is 0.


The friskily announce items from multiple object lists
rule is listed instead of the announce items from multiple
object lists rule in the action-processing rules.


This is the friskily announce items from multiple object
lists rule:
    if taking:
        if the current item from the multiple object list is not
nothing:
            increment the number of takes this turn;
            say "[if number of takes this turn is 1]First
[otherwise if the number of takes this turn is 2]And then
[otherwise if the number of takes this turn is 3]And I
suppose also [otherwise if the number of takes this turn
is 7]And on we wearily go with [otherwise if the number
of takes this turn is 9]Oh, and not forgetting
[otherwise]And [end if][the current item from the multiple
object list]: [run paragraph on]";
    otherwise:
        if the current item from the multiple object list is not
nothing, say "[current item from the multiple object list]:
[run paragraph on]".
```

Rule for deciding whether all includes the person asked: it does not.
Rule for deciding whether all includes a person when taking: it does not.

Section 2 - The Scenario

The Palm Chamber is a room. Sarissa is a woman in the Palm Chamber.

The Palm Chamber contains a bottle of ink, a quill pen, a tangerine, a bunch of grapes, a length of silken rope, some perfume, a cake of incense, a fitted leather bodice, a sapphire anklet, an illustrated novel, a whip, and a heavy iron key.

A persuasion rule for asking Sarissa to try taking the key:
    say "Sarissa nervously demurs, knowing that it is forbidden.";
    persuasion fails.

A persuasion rule: persuasion succeeds.

Test me with "take all / drop all / sarissa, take all".

Chapter 20: Advanced Text

ohs; *§20.4. Upper and lower case letters*; *§20.5. Matching and exactly matching*; *§20.6. Regular expres*

Contents of *Writing with Inform*
Chapter 19: Rulebooks
Chapter 21: Lists
Indexes of the examples

§20.1. Changing texts

So far, we have dealt with text as something which comes in little packets: we have printed it out, read it in from the keyboard, and compared it with other text. But we have never tried to open the packets and get at the contents, letter by letter, or to make any alterations, or look for certain combinations of letters. These tricks are surprisingly seldom needed - a surprise, that is, given that everything Inform does is textual - but they are in fact open to us. For example:

```
if character number 1 in "[time of day]" is "1", ...
```

will be true at, for example, 11:30 PM and 1:22 AM, but not at 3:15 PM. What happens here is that Inform expands the time of day into a text, say "11:30 PM", then extracts the first character, say "1", and tests it.

Until 2012, Inform had two kinds of text - plain "text", and "indexed text" - but there's now only "text", which has all of the abilities of both.

Start of Chapter 20: Advanced Text

Back to Chapter 19: Rulebooks: §19.16. The Laws for Sorting Rulebooks

Onward to §20.2. Memory limitations

---

§20.2. Memory limitations

Inform creates "story files" for very small virtual computers (capable of running on phones, for instance) where memory is tight. If we create a number variable and keep on adding 1 to it, the value simply gets bigger. But if we make some text and keep on adding a letter "x" to it, the text takes up more and more space, growing into longer and longer runs of "x"s until there is no more space to hold it.

The following warnings are rather like the tiny print about side-effects on medicine bottles: that is, we mostly ignore them, and if the drugs should kill us, well, at least we have the consolation of knowing we were warned. There are basically three limitations on text:

(1) An amount of memory has to be set aside for text (and other flexible-sized data), and Inform guesses the amount needed. Story files using the Glulx format (see the Settings panel) are able to increase this as necessary in play, so there's no problem if the guess was wrong. But Z-machine story files are stuck with whatever amount of memory was initially chosen.

That choice can be increased with a use option, like so:

```
Use dynamic memory allocation of at least 16384.
```

Inform raises its estimate of the amount needed to ensure that this amount is always at least its own guess, and also at least any amount declared like this. (And then it rounds up to the nearest power of 2, as it happens.) The default value of "dynamic memory allocation" is 8192. In practice, this use option isn't needed much, though, because any story needing large amounts of dynamic memory will likely be on Glulx in any case.

(2) Text has a maximum length. This maximum is normally 1000 characters, which ought to be plenty, but can be raised by sentences such as:

```
Use maximum text length of at least 2000.
```

What happens if this is broken, that is, if we try to use text overrunning this length? The Z-machine may simply crash, so if there is any chance that any single text may grow unpredictably large, Glulx should always be used. On Glulx, overrunning text is truncated safely, except that under Glulx 3.1.0 or better the story file will try to use dynamic memory allocation to expand the limit as needed to avoid truncation. (Testing shows that text is slow to manipulate once it grows beyond about 20,000 characters in length, but this is not really surprising.)

(3) Under the Z-machine, text may only contain characters from the so-called "ZSCII" character set - standard numbers, letters, punctuation marks and the commonest West European accented letters. Anything more exotic is likely to be flattened into a question mark "?". Under Glulx, any character can be used.

All of this makes the Z-machine sound very inferior, for text purposes. But note that Z can handle all of the examples in this chapter perfectly happily.

Start of Chapter 20: Advanced Text
Back to §20.1. Changing texts
Onward to §20.3. Characters, words, punctuated words, unpunctuated words, lines, paragraphs

§20.3. Characters, words, punctuated words, unpunctuated words, lines, paragraphs

Inform can get at the contents of text in a variety of ways. The lowest-level is by character - a character is a letter, digit, punctuation symbol, space or other letter-form. (We use the term "character" rather than "letter" because otherwise we would have to call "5" a letter, and so on.) Characters number upwards from 1: character number 1, to repeat that, starts the text. We can get the Nth character with:

character number (number) **in** (text) **... text**

This phrase produces the Nth character from the text, counting from 1. Characters include letters, digits, punctuation symbols, spaces or other letter-forms. Example:

```
character number 8 in "numberless projects of social reform"
```

produces "e". If the index is less than 1 or more than the length of the text, the result is an empty text, "".

The maximum character number varies with the current length of the text, and can be evaluated as:

number of characters in (text) **... number**

This phrase produces the number of characters from the text. Characters include letters, digits, punctuation symbols, spaces or other letter-forms.

Examples:

```
number of characters in "War and Peace"
number of characters in ""
```

produce 13 and 0 respectively.

We can also use the adjective "empty":

```
if the description of the location is empty, ...
```

The empty text, "", is the only one with 0 characters.

We can also extract the contents by word, again numbered from 1. Thus:

word number (number) **in** (text) **... text**

This phrase produces the Nth word from the text, counting from 1. Words for this purpose are what's left after breaking the text up at punctuation or spacing (spaces, line breaks, paragraph breaks) and then removing that punctuation or spacing. Example:

```
word number 3 in "ice-hot, don't you think?"
```

produces "don't". If the index is less than 1 or more than the number of words in the text, the result is an empty text, "".

number of words in (text) **... number**

This phrase produces the number of words from the text. Words for this purpose are what's left after breaking the text up at punctuation or spacing (spaces, line breaks, paragraph breaks) and then removing that punctuation or spacing. Example:

```
number of words in "ice-hot, don't you think?"
```

produces 5.

Note that the contraction apostrophe in "don't" doesn't count as punctuation. Because this is not always quite what we want, Inform offers two variations:

punctuated word number (number) **in** (text) **... text**

This phrase produces the Nth word from the text, counting from 1. Words for this purpose are what's left after breaking the text up at punctuation or spacing (spaces, line breaks, paragraph breaks) and then removing the spacing, but leaving the punctuation as independent words. Example:

```
punctuated word number 2 in "ice-hot, don't you think?"
```

produces "-". The punctuated words here are "ice", "-", "hot", ",", "don't", "you", "think", "?". If two or more punctuation marks are adjacent, they are counted as different words, except for runs of dashes or periods: thus ",," has two punctuated words, but "--" and "..." have only one each. If the index is less than 1 or more than the number of punctuated words in the text, the result is an empty text, "".

number of punctuated words in (text) **... number**

This phrase produces the number of words from the text. Words for this purpose are what's left after breaking the text up at punctuation or spacing (spaces, line breaks, paragraph breaks) and then removing the spacing, but leaving the punctuation as independent words. Example:

```
number of punctuated words in "ice-hot, don't you think?"
```

produces 8; see if you can find them all.

unpunctuated word number (number) **in** (text) **... text**

This phrase produces the Nth word from the text, counting from 1. Words for this purpose are what's left after breaking the text up at spacing (spaces, line breaks, paragraph breaks) but including all punctuation as if it were part of

the spelling of the words it joins to. Example:

unpunctuated word number 1 in "ice-hot, don't you think?"

produces "ice-hot,". The unpunctuated words in "ice-hot, don't you think?" are "ice-hot,", "don't", "you", "think?". If the index is less than 1 or more than the number of punctuated words in the text, the result is an empty text, "".

number of unpunctuated words in (text) **... number**

This phrase produces the number of words from the text. Words for this purpose are what's left after breaking the text up at spacing (spaces, line breaks, paragraph breaks) but including all punctuation as if it were part of the spelling of the words it joins to. Example:

number of unpunctuated words in "ice-hot, don't you think?"

produces just 4.

Finally, on the larger scale still, we also have:

line number (number) **in** (text) **... text**

This phrase produces the Nth line from the text, counting from 1. Unless explicit use is made of line-breaking, lines and paragraphs will be the same - it doesn't refer to lines as visible on screen, because we have no way of knowing what size screen the player might have.

number of lines in (text) **... number**

This phrase produces the number of lines in the text. Unless explicit use is made of line-breaking, lines and paragraphs will be the same - it doesn't refer to lines as visible on screen, because we have no way of knowing what size screen the player might have. Example: the number of lines in

> "Sensational news just in![paragraph break]The Martians have invaded Miranda.[line break](One of the moons of Uranus, that is.)"

is 3.

paragraph number (number) **in** (text) **... text**

This phrase produces the Nth paragraph from the text, counting from 1.

number of paragraphs in (text) **... number**

This phrase produces the number of paragraphs in the text. Example: the number of paragraphs in

> "Sensational news just in![paragraph break]The Martians have invaded Miranda.[line break](One of the moons of Uranus, that is.)"

is 2.

(Attempting to make large enough texts to have a serious paragraph count is slightly risky if there is not much memory to play with, as on the Z-machine. But the facilities do exist.)

Start of Chapter 20: Advanced Text
Back to §20.2. Memory limitations
Onward to §20.4. Upper and lower case letters

§20.4. Upper and lower case letters

In most European languages the same letters can appear in two forms: as capitals, like "X", mainly used to mark a name or the start of a sentence; or in their ordinary less prominent form, like "x". These forms are called upper and lower case because, historically, typesetters kept lead castings of letters in two wooden cases, one above

the other on the workbench. Lower case letters were in the lower box closer to hand, being more often needed.

Human languages are complicated. Not every lower case letter has an upper case partner: ordinal markers in Hispanic languages don't, for instance, and the German "ß" is never used in upper case. Sometimes two different lower case letters have the same upper case form: "ς" and "σ", two versions of the Greek sigma, both capitalise to "Σ". Inform follows the international Unicode standard in coping with all this.

We can test whether text is in either case like so:

if (text) **is in lower case:**

This condition is true if every character in the text is a lower case letter. Examples: this is true for "wax", but false for "wax seal" or "eZ mOnEy".

if (text) **is in upper case:**

This condition is true if every character in the text is in upper case. Examples: this is true for "BEESWAX", but false for "ROOM 101".

We can change the casing of text using:

(text) **in lower case ... text**

This phrase produces a new version of the given text, but with all upper case letters reduced to lower case. Example: "a ticket to Tromsø via Østfold" becomes

"a ticket to tromsø via østfold"

(text) **in upper case ... text**

This phrase produces a new version of the given text, but with all upper case letters reduced to lower case. Example: "a ticket to Tromsø via Østfold" becomes

"A TICKET TO TROMSØ VIA ØSTFOLD"

(text) **in title case ... text**

This phrase produces a new version of the given text, but with casing of words changed to title casing: this capitalises the first letter of each word, and lowers the rest. Example: "a ticket to Tromsø via Østfold" becomes

"A Ticket To Tromsø Via Østfold"

(text) **in sentence case ... text**

This phrase produces a new version of the given text, but with casing of words changed to sentence casing: this capitalises the first letter of each sentence and reduces the rest to lower case. Example: "a ticket to Tromsø via Østfold" becomes

"A ticket to tromsø via østfold"

Accents are preserved in case changes. So (if we are using Glulx and have Unicode available) title case can turn Aristophanes' discomfortingly lower-case lines

ἐξ οὗ γὰρ ἡμᾶς προὔδοσαν μιλήσιοι,
οὐκ εἶδον οὐδ᾽ ὄλισβον ὀκτωδάκτυλον,
ὃς ἦν ἂν ἡμῖν σκυτίνη "πικουρία

by raising them proudly up like so:

Ἐξ Οὗ Γὰρ Ἡμᾶς Προὔδοσαν Μιλήσιοι,
Οὐκ Εἶδον Οὐδ᾽ Ὄλισβον Ὀκτωδάκτυλον,
Ὃς Ἦν Ἂν Ἡμῖν Σκυτίνη "Πικουρία.

Title and sentence casing can only be approximate if done by computer. Inform looks at the letters, but is blind to the words and sentences they make up. (Note the way sentence casing did not realise "Tromsø" and "Østfold" were proper nouns.) If asked

to put the name "MCKAY" into title casing, Inform will opt for "Mckay", not recognising this as the Scottish patronymic surname "McKay". Given "baym dnieper", the title of David Bergelson's great Yiddish novel of 1932, it will opt for "BAYM DNIEPER": but properly speaking Yiddish does not have upper case lettering at all, though nowadays it is sometimes printed as if it did. And conventions are very variable about which words should be capitalised in titles: English publishers mostly agree that connectives, articles and prepositions should be in lower case, but in France almost anything goes, with Académie Française rules giving way to avant-garde book design. In short, we cannot rely on Inform's title casing to produce a result which a human reader will always think perfect.

This discussion has all been about how Inform prints, not about how it reads commands from the keyboard, because the latter is done case-insensitively. The virtual machines for which Inform creates programs normally flatten all command input to lower case, and in any case Understand comparison ignores casing. Thus

```
Understand "mckay" as the Highland Piper.
```

means that "examine McKay", "examine MCKAY", "examine mckay", and so forth are all equivalent. The text of the player's command probably doesn't preserve the original casing typed in any event.

One more caution, though it will affect hardly anyone. For projects using the Z-machine, only a restricted character set is available in texts: for more, we must use Glulx. A mad anomaly of ZSCII, the Z-machine character set, is that it contains the lower case letter "ÿ" but not its upper case form "Ÿ", so that

```
"ÿ" in upper case
```

produces "Ÿ" in Glulx but "ÿ" in the Z-machine. This will come as a blow to Queensrÿche fans, but in all other respects any result on the Z-machine should agree with its counterpart on Glulx.

Start of Chapter 20: Advanced Text
Back to §20.3. Characters, words, punctuated words, unpunctuated words, lines, paragraphs
Onward to §20.5. Matching and exactly matching

Example 412: **Capital City**
To arrange that the location

information normally given on the left-hand side of the status line appears in block capitals.

Example 413: ★ **Rocket Man**
Using case changes on any text produced by a "to say..." phrase.

---

§20.5. Matching and exactly matching

Up to now, we have only been able to judge two texts by seeing if they are equal, but we can now ask more subtle questions.

if (text) **matches the text** (text)**:**

This condition is true if the second text occurs anywhere inside the first. Examples:

```
if "[score]" matches the text "3", ...
```

tests whether the digit 3 occurs anywhere in the score, as printed out; and

```
if the printed name of the location matches the text "the", ...
```

tests to see whether "the" can be found anywhere in the current room's name. Note that the location "Smotheringly Hot Jungle" would pass this test - it's there if you look. On the other hand, "The Orangery" would not, because "The" does not match against "the". We can get around this in a variety of ways, one of which is to tell Inform to be insensitive to the case (upper or lower) of letters:

```
if the printed name of the location matches the text "the",
case insensitively: ...
```

if (text) **exactly matches the text** (text)**:**

This condition is true if the second text matches the first, starting at the beginning and finishing at the end. This appears to be the same as testing if one is equal to the other, but that's not quite true: for example,

```
if "[score]" exactly matches the text "[best score]", ...
```

is true if the score and best score currently print out as the same text, which will be true if they are currently equal as numbers; but

```
if "[score]" is "[best score]", ...
```

is never true - these are different texts, even if they sometimes look the same.

In the next section we shall see that "matches" and "exactly matches" can do much more than the simple text matching demonstrated above.

We can also see how many times something matches:

number of times (text) **matches the text** (text) **... number**

This produces the number of times the second text occurs within the first. The matches are not allowed to overlap. Example:

```
number of times "pell-mell sally" matches the text "ll" = 3
number of times "xyzzy" matches the text "Z" = 0
number of times "xyzzy" matches the text "Z", case
insensitively = 2
number of times "aaaaaaaa" matches the text "aaaa" = 2
```

There's no "number of times WHATEVER exactly matches the text FIND" phrase since this is by definition going to have to be 0 or 1.

Start of Chapter 20: Advanced Text

Back to §20.4. Upper and lower case letters

Onward to §20.6. Regular

§20.6. Regular expression matching

When playing around with text, we tend to get into longer and trickier wrangles of matching - we find that we want to look not for simple text like "gold", but for "gold" used only as a separate word, or for a date in YYYY-MM-DD format, or for a seemingly endless range of other possibilities. What we need is not just for Inform to provide a highly flexible matching program, but also a good notation in which to describe what we want.

Fortunately, such a notation already exists. This is the "regular expression" notation, named for a 1950s mathematical model by the logician Stephen Kleene, applied to computing in the late 60s by Ken Thompson, borrowed almost at once by the early Unix tools of the 70s, and developed further by Henry Spencer in the 80s and Philip Hazel in the 90s. The glue holding the Internet together - the Apache web-server, the scripting languages Perl and Python, and so forth - makes indispensable use of regular expressions.

As might be expected from the previous section, we simply have to describe the FIND text as "regular expression" rather than "text" and then the same facilities are available:

if (text) **matches the regular expression** (text)**:**

This condition is true if any contiguous part of the text can be matched against the given regular expression. Examples:

```
if "taramasalata" matches the regular expression "a.*l", ...
```

is true, since this looks for a part of "taramasalata" which begins with "a", continues with any number of characters, and finishes with "l"; so it matches "aramasal". (Not "asal", because it gets the makes the leftmost match it can.) The option "case insensitively" causes lower and upper case letters to be treated as equivalent.

if (text) **exactly matches the regular expression** (text)**:**

This condition is true if the whole text (starting from the beginning and finishing at the end) can be matched against the given regular expression. The option "case insensitively" causes lower and upper case letters to be

treated as equivalent.

And once again:

number of times (text) **matches the regular expression** (text) **... number**

This produces the number of times that contiguous pieces of the text can be matched against the regular expression, without allowing them to overlap.

Since a regular expression can match quite a variety of possibilities (for instance "b\w +t" could match "boast", "boat", "bonnet" and so on), it's sometimes useful to find what the match actually was:

text matching regular expression ... text

This phrase is only meaningful immediately after a successful match of a regular expression against text, and it produces the text which matched. Example:

```
if "taramasalata" matches the regular expression "m.*l":
    say "[text matching regular expression].";
```

says "masal."

Perhaps fairly, perhaps not, regular expressions have a reputation for being inscrutable. The basic idea is that although alphanumeric characters (letters, numbers and spaces) mean just what they look like, punctuation characters are commands with sometimes dramatic effects. Thus:

```
if WHATEVER matches the regular expression "fish", ...
if WHATEVER matches the regular expression "f.*h", ...
```

behave very differently. The first is just like matching the text "fish", but the second matches on any sequence of characters starting with an "f" and ending with an "h". This is not at all obvious at first sight: reading regular expressions is a skill which must be learned, like reading a musical score. A really complex regular expression can look like a soup of punctuation and even an expert will blink for a few minutes before telling you what it does - but a beginner can pick up the basics very quickly. Newcomers might like to try out and become comfortable with the features a few at a

time, reading down the following list.

1. Golden rule. Don't try to remember all the characters with weird effects. Instead, if you actually mean any symbol other than a letter, digit or space to be taken literally, place a backslash "\" in front of it. For instance, matching the regular expression

```
"\*A\* of the Galactic Patrol"
```

is the same as matching the text "*A* of the Galactic Patrol", because the asterisks are robbed of their normal powers. This includes backslash itself: "\\" means a literal backslash. (Don't backslash letters or digits - that turns out to have a meaning all its own, but anyway, there is never any need.)

2. Alternatives. The vertical stroke "|" - not a letter I or L, nor the digit 1 - divides alternatives. Thus

```
"the fish|fowl|crawling thing"
```

is the same as saying match "the fish", or "fowl", or "crawling thing".

3. Dividing with brackets. Round brackets "(" and ")" group parts of the expression together.

```
"the (fish|fowl|crawling thing) in question"
```

is the same as saying match "the fish in question", or "the fowl in question", or "the crawling thing in question". Note that the "|" ranges outwards only as far as the group it is in.

4. Any character. The period "." means any single character. So

```
"a...z"
```

matches on any sequence of five characters so long as the first is "a" and the last is "z".

5. Character alternatives. The angle brackets "<" and ">" are a more concise way of specifying alternatives for a single character. Thus

```
"b<aeiou>b"
```

matches on "bab", "beb", "bib", "bob" or "bub", but not "baob" or "beeb" - any single character within the angle brackets is accepted. Beginning the range with "^" means "any single character so long as it is not one of these": thus

```
"b<^aeiou>b"
```

matches on "blb" but not "bab", "beb", etc., nor on "blob" or "bb". Because long runs like this can be a little tiresome, we are also allowed to use "-" to indicate whole ranges. Thus

```
"b<a-z>b"
```

matches a "b", then any lower case English letter, then another "b".

In traditional regular expression language, square brackets rather than angle brackets are used for character ranges. In fact Inform does understand this notation if there are actual square brackets "[" and "]" in the pattern text, but in practice this would be tiresome to achieve, since Inform uses those to achieve text substitutions. So Inform allows "b<a-z>b" rather than making us type something like

```
"b[bracket]a-z[close bracket]b"
```

to create the text "b[a-z]b".

6. Popular character ranges. The range "<0-9>", matching any decimal digit, is needed so often that it has an abbreviation: "\d". Thus

```
"\d\d\d\d-\d\d-\d\d"
```

matches, say, "2006-12-03". Similarly, "\s" means "any spacing character" - a space, tab or line break. "\p" is a punctuation character, in the same sense used for word division in the previous section: it actually matches any of

```
. , ! ? - / " : ; ( ) [ ] { }
```

"\w" means "any character appearing in a word", and Inform defines it as anything not matching "\s" or "\p".

"\l" and "\u" match lower and upper case letters, respectively. These are much stronger than "<a-z>" and "<A-Z>", since they use the complete definition in the Unicode 4.0.0 standard, so that letter-forms from all languages are catered for: for example "δ" matches "\l" and "Δ" matches "\u".

The reverse of these is achieved by capitalising the letter. So "\D" means "anything not a digit", "\P" means "anything not punctuation", "\W" means "anything not a word character", "\L" means "anything not a lower case letter" and so on.

7. Positional restrictions. The notation "^" does not match anything, as such, but instead requires that we be positioned at the start of the text. Thus

```
"^fish"
```

matches only "fish" at the start of the text, not occurring anywhere later on. Similarly, "$" requires that the position be the end of the text. So

```
"fish$"
```

matches only if the last four characters are "fish". Matching "^fish$" is the same thing as what Inform calls exactly matching "fish".

Another useful notation is "\b", which matches a word boundary: that is, it matches no actual text, but requires the position to be a junction between a word character and a non-word character (a "\w" and a "\W") or vice versa. Thus

```
"\bfish\b"
```

matches "fish" in "some fish" and also "some fish, please!", but not in "shellfish". (The regular expression "\w*fish\b" catches all words ending in "fish", as we will see below.) As usual, the capitalised version "\B" negates this, and means "not at a word boundary".

8. Line break and tab. The notations "\n" and "\t" are used for a line break ("n" for "new line") and tab, respectively. Tabs normally do not occur in Inform strings, but can do when reading from files. It makes no sense to reverse these, so "\N" and "\T" produce errors.

9. Repetition. Placing a number in braces "{" and "}" after something says that it should be repeated that many times. Thus

```
"ax{25}"
```

matches only on "axxxxxxxxxxxxxxxxxxxxxxxxx". More usefully, perhaps, we can specify a range of the number of repetitions:

```
"ax{2,6}"
```

matches only on "axx", "axxx", "axxxx", "axxxxx", "axxxxxx". And we can leave the top end open: "ax{2,}" means "a" followed by at least two "x"s.

Note that the braces attach only to most recent thing - so "ax{2}" means "a" followed by two of "x" - but, as always, we can use grouping brackets to change that. So "(ax){2,}" matches "axax", "axaxax", "axaxaxax",...

(It's probably best not to use Inform to try to match the human genome against "<acgt>{3000000000}", but one of the most important practical uses of regular expression matching in science is in treating DNA as a string of nucleotides represented by the letters "a", "c", "g", "t", and looking for patterns.)

10. Popular repetitions. Three cases are so often needed that they have standard short forms:

"{0,1}", which means 0 or 1 repetition of something - in other words, doesn't so much repeat it as make it optional - is written "?". Thus "ax?y" matches only on "ay" or "axy".

"{0,}", which means 0 or more repetitions - in other words, any number at all - is written "*". Thus "ax*y" matches on "ay", "axy", "axxy", "axxxy", ... and the omnivorous ".*" - which means "anything, any number of times" - matches absolutely every text. (Perhaps unexpectedly, replacing ".*" in a text with "X" will produce "XX", not "X", because the ".*" first matches the text, then matches the empty gap at the end. To match the entire text just once, try "^.*$".)

"{1,}", which means 1 or more repetitions, is written "+". So "\d+" matches any run of digits, for instance.

11. Greedy vs lazy. Once we allow things to repeat an unknown number of times, we run into an ambiguity. Sure, "\d+" matches the text "16339b". But does it look only as far as the "1", then reason that it now has one or more digits in a row, and stop? Or does it run onward devouring digits until it can do so no longer, so matching the "16339" part? These two strategies are called "lazy" and "greedy" respectively.

Do we care? Well, the strategy used makes no difference to whether there is a match, but it does affect what part of the text is matched, and the number of matches there are. Unless we mark for it, all repetitions are greedy. Usually this is good, but it means that, for instance,

```
"-.+-"
```

applied to "-alpha- -beta- -gamma-" will match the whole text, because ".+" picks up all of "alpha- -beta- -gamma". To get around this, we can mark any of the repetition operators as lazy by adding a question mark "?". Thus:

```
"-.+?-"
```

applied to "-alpha- -beta- -gamma-" matches three times, producing "-alpha-" then "-beta-" then "-gamma-".

A logical but sometimes confusing consequence is that a doubled question mark "??" means "repeat 0 or 1 times, but prefer 0 matches to 1 if both are possibilities": whereas a single question mark "?", being greedy, means "repeat 0 or 1 times, but prefer 1 match to 0 if both are possibilities".

12. Numbered groups. We have already seen that round brackets are useful to clump together parts of the regular expression - to choose within them, or repeat them. In fact, Inform numbers these from 1 upwards as they are used from left to right, and we can subsequently refer back to their contents with the notation "\1", "\2", ... After a successful match, we can find the results of these subexpressions with:

> **text matching subexpression** (number) **... text**
>
> This phrase is only meaningful immediately after a successful match of a regular expression against text, and it produces the text which matched. The number must be from 1 to 9, and must correspond to one of the bracketed groups in the expression just matched. Example: after
>
> ```
> if "taramasalata" matches the regular expression "a(r.*l)a(.)":
> ```
>
> the "text matching regular expression" is "aramasalat", the "text matching subexpression 1" is "ramasal", and "text matching subexpression 2" is "t".

For instance:

```
"(\w)\w*\1"
```

matches any run of two or more word-characters, subject to the restriction that the last one has to be the same as the first - so it matches "xerox" but not "alphabet".

When Inform matches this against "xerox", first it matches the initial "x" against the group "(\w)". It then matches "\w*" ("any number of word-characters") against "ero", so that the "*" runs up to 3 repetitions. It then matches "\1" against the final "x", because "\1" requires it to match against whatever last matched in sub-expression 1 - which was an "x".

Numbered groups allow wicked tricks in matching, it's true, but really come into their own when it comes to replacing - as we shall see.

13. Switching case sensitivity on and off. The special notations "(?i)" and "(?-i)" switch sensitivity to upper vs. lower case off and on, mid-expression. Thus "a(?i)bcd(?-i)e" matches "abcde", "aBcDe", etc., but not "Abcde" or "abcdE".

14. Groups with special meanings. This is the last of the special syntaxes: but it's a doozy. A round-bracketed group can be marked to behave in a special way by following the open bracket by a symbol with a special meaning. Groups like this have no number and are not counted as part of \1, \2, and so forth - they are intended not to gather up material but to have some effect of their own.

"(# ...)"

Is a comment, that is, causes the group to do nothing and match against anything.

"(?= ...)"

Is a lookahead: it is a form of positional requirement, like "\b" or "^", but one which requires that the text ahead of us matches whatever is in the brackets. (It doesn't consume that text - only checks to see that it's there.) For instance "\w+(?=;)" matches a word followed by a semicolon, but does not match the semicolon itself.

"(?! ...)"

Is the same but negated: it requires that the text ahead of us does not match the material given. For instance, "a+(?!z)" matches any run of "a"s not followed by a "z".

"(?<= ...)" and "(?<! ...)"

Are the same but looking behind (hence the "<"), not forward. These are restricted to cases where Inform can determine that the material to be matched has a definite known width. For instance, "(?<!shell)fish" matches any "fish" not occurring in "shellfish".

"(> ...)"

Is a possessive, that is, causes the material to be matched and, once matched, never lets go. No matter what subsequently turns out to be convenient, it will never change its match. For instance, "\d+8" matches against "768" because Inform realises that "\d+" cannot be allowed to eat the "8" if there is to be a match, and stops it. But "(>\d+)8" does not match against "768" because now the "\d+", which initially eats "768", is possessive and refuses to give up the "8" once taken.

"(?(1)...)" and "(?(1)...|...)"

Are conditionals. These require us to match the material given if \1 has successfully matched already; in the second version, the material after the "|" must be matched if \1 has not successfully matched yet. And the same for 2, 3, ..., 9, of course.

Finally, conditionals can also use lookaheads or lookbehinds as their conditions. So for instance:

"(?(?=\d)\d\d\d\d|AY-\d\d\d\d)"

means if you start with a digit, match four digits; otherwise match "AY-" followed by four digits. There are easier ways to do this, of course, but the really juicy uses of conditionals are only borderline legible and make poor examples - perhaps this is telling us something.

Start of Chapter 20: Advanced Text

Back to §20.5. Matching and exactly matching

Onward to §20.7. Making new text with text substitutions

Example 414: ★ **Alpha**
Creating a beta-testing command that matches any line starting with punctuation.

Example 415: ★ **About Inform's regular expression support**
Some footnotes on Inform's

regular expressions, and how they compare to those of other programming languages.

§20.7. Making new text with text substitutions

Substitutions are most often used just for printing, like so:

```
say "The clock reads [time of day].";
```

But they can also produce text which can be stored up or used in other ways. For example, defining

```
To decide what text is (T - text) doubled:
    decide on "[T][T]".
```

makes

```
let the Gerard Kenny reference be "NewYork" doubled;
```

set this temporary variable to "NewYorkNewYork".

There is, however, a subtlety here. A text with a substitution in it, like:

```
"The clock reads [time of day]."
```

is always waiting to be substituted, that is, to become something like:

```
"The clock reads 11:12 AM."
```

If all we do with text is to print it, there's nothing to worry about. But if we're storing it up, especially for multiple turns, there are ambiguities. For example, suppose we're changing the look of the black status line bar at the top of the text window:

```
now the left hand status line is "[time of day]";
```

Just copying "[time of day]" to the "left hand status line" variable doesn't make it substitute - which is just as well, or the top of the screen would perpetually show "9:00 AM".

On the other hand, looking back at the phrase example:

```
To decide what text is (T - text) doubled:
   decide on "[T][T]".
```

"[T][T]" is substituted immediately it's formed. That's also a good thing, because "T" loses its meaning the moment the phrase finishes, which would make "[T][T]" meaningless anywhere else.

What's going on here is this: Inform substitutes text immediately if it contains references to a temporary value such as "T", and otherwise only if it needs to access the contents. This is why "[time of day]" isn't substituted until we need to print it out (or, say, access the third character): "time of day" is a value which always exists, not a temporary one.

Using the adjectives "substituted" and "unsubstituted", it's always possible to test whether a given text is in either state, should this ever be useful. For example,

```
now the left hand status line is "[time of day]";
if the left hand status line is unsubstituted, say "Yes!";
```

will say "Yes!": the LHSL is like a bomb waiting to go off. Speaking of which:

```
The player is holding a temporal bomb.


When play begins:
   now the left hand status line is "Clock reads: [time of day]".


After dropping the temporal bomb:
   now the left hand status line is the substituted form of the left hand
status line;
   say "Time itself is now broken. Well done."
```

This is making use of:

substituted form of (text) **... text**

This takes a text and makes substitution occur immediately. For example,

```
substituted form of "time of death, [time of day]"
```

produces something like "time of death, 9:15 AM" rather than "time of death, [time of day]". It's entirely legal to apply this to text which never had any substitutions in, so

```
substituted form of "balloon"
```

produces "balloon".

Note that there's no analogous phrase for "unsubstituted form of...", because once text has substituted, there's no way to go back.

Start of Chapter 20: Advanced Text
Back to §20.6. Regular expression matching
Onward to §20.8. Replacements

Example 416: ★ **Identity Theft** Allowing the player to enter a name to be used for the player character during the game.

Example 417: ★ **Mirror, Mirror** The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

Example 418: ★★ **The Cow Exonerated** Creating a class of matches that burn for a time and

then go out, with elegant reporting when several matches go out at once.

§20.8. Replacements

Suppose V is a text which varies - perhaps a property of something, or a variable defined everywhere, or a temporary "let"-named value. How do we change its contents? The easiest way is simply to assign text to it. Thus:

```
let V be "It is now [the time of the day in words]."
```

And, for instance,

```
let V be "[V]!"
```

adds an exclamation mark at the end of V.

Otherwise, it is more useful (also a little faster) to modify V by changing its characters, words and so on. Thus:

replace character number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the Nth character, counting from 1. Example:

```
let V be "mope";
replace character number 3 in V with "lecul";
say V;
```

says "molecule".

replace word number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the Nth word, counting from 1, and dividing words at spacing or punctuation.

Example:

```
let V be "Does the well run dry?";
replace word number 3 in V with "jogger";
say V;
```

says "Does the jogger run dry?".

replace punctuated word number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the Nth word, counting from 1, and dividing words at spacing, counting punctuation runs as words in their own right. Example:

```
let V be "Frankly, yes, I agree.";
replace punctuated word number 2 in V with ":";
say V;
```

says "Frankly: yes, I agree.".

replace unpunctuated word number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the Nth word, counting from 1, and dividing words at spacing, counting punctuation as part of a word just as if it were lettering. Example:

```
let V be "Frankly, yes, I agree.";
replace unpunctuated word number 2 in V with "of course";
say V;
```

says "Frankly, of course I agree.".

replace line number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the

Nth line, counting from 1. Lines are divided by paragraph or line breaks.

replace paragraph number (number) **in** (text) **with** (text)

This phrase acts on the named text by placing the given text in place of the Nth paragraph, counting from 1.

Last, but not least, we can replace text wherever it occurs:

replace the text (text) **in** (text) **with** (text)

This phrase acts on the named text by searching and replacing, as many non-overlapping times as possible. Example:

```
replace the text "a" in V with "z"
```

changes every lower-case "a" to "z": the same thing done with the "case insensitively" option would change each "a" or "A" to "z".

All very well for letters, but it can be unfortunate to try

```
replace the text "Bob" in V with "Robert"
```

if V happens to contain, say "The Olympic Bobsleigh Team": it would become "The Olympic Robertsleigh Team". What we want, of course, is for Bob to become Robert only when it's a whole word. We can get that with:

replace the word (text) **in** (text) **with** (text)

This phrase acts on the named text by searching and replacing, as many non-overlapping times as possible, where the search text must occur as a whole word. Example:

```
replace the word "Bob" in V with "Robert"
```

changes "Bob got on the Bobsleigh" to "Robert got on the Bobsleigh".

replace the punctuated word (text) **in** (text) **with** (text)

This phrase acts on the named text by searching and replacing, as many non-overlapping times as possible, where the search text must occur as a whole word or run of punctuation.

But these are all just special cases of the grand-daddy of all replacement phrases:

replace the regular expression (text) **in** (text) **with** (text)

This phrase acts on the named text by matching the regular expression and replacing anything which fits it, as many non-overlapping times as possible. Example:

```
replace the regular expression "\d+" in V with "..."
```

changes "The Battle of Waterloo, 1815, rivalled Trafalgar, 1805" to "The Battle of Waterloo, ..., rivalled Trafalgar, ...". The "case insensitively" causes lower and upper case letters to be treated as if the same letter.

When replacing a regular expression, the replacement text also has a few special meanings (though, thankfully, many fewer than for the expression itself). Once again "\n" and "\t" can be used for line break and tab characters, and "\\" must be used for an actual backslash. But, very usefully, "\1" to "\9" expand as the contents of groups numbered 1 to 9, and "\0" to the exact text matched. So:

```
replace the regular expression "\d+" in V with "roughly \0"
```

adds the word "roughly" in front of any run of digits in V, because \0 becomes in turn whichever run of digits matched. And

```
replace the regular expression "(\w+) (.*)" in V with "\2, \1"
```

performs the transformation "Frank Booth" to "Booth, Frank".

Finally, prefixing the number by "l" or "u" forces the text it represents into lower or upper case, respectively. For instance:

```
replace the regular expression "\b(\w)(\w*)" in X with "\u1\l2";
```

changes the casing of X to "title casing", where each individual word is capitalised. (This is a little slow on large texts, since so many matches and replacements are made: it's more efficient to use the official phrases for changing case.)

Start of Chapter 20: Advanced Text

Back to §20.7. Making new text with text substitutions

Onward to §20.9. Summary of regular expression notation

Example 419: ★ **Blackout** Filtering the names of rooms printed while in darkness.

Example 420: ★ **Fido** A dog the player can name and un-name at will.

Example 421: ★ **Igpay Atinlay** A pig Latin filter for the player's commands.

Example 422: ★★ **Mr. Burns' Repast** Letting the player guess types for an unidentifiable fish.

Example 423: ★★ **Northstar** Making Inform understand ASK JOSH TO TAKE INVENTORY as JOSH, TAKE INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

Example 424: ★★★ **Cave-**

troll Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

---

§20.9. Summary of regular expression notation

MATCHING

Positional restrictions

```
^      Matches (accepting no text) only at the start of the text
$      Matches (accepting no text) only at the end of the text
\b     Word boundary: matches at either end of text or between a \w and a \W
\B     Matches anywhere where \b does not match
```

Backslashed character classes

```
\char  If char is other than a-z, A-Z, 0-9 or space, matches that literal char
       For example, this matches literal backslash "\"
\n     Matches literal line break character
\t     Matches literal tab character (but use this only with external files)
```

```
\d     Matches any single digit
\l     Matches any lower case letter (by Unicode 4.0.0 definition)
\p     Matches any single punctuation mark: . , ! ? - / " : ; ( ) [ ] { }
\s     Matches any single spacing character (space, line break, tab)
\u     Matches any upper case letter (by Unicode 4.0.0 definition)
\w     Matches any single word character (neither \p nor \s)
```

```
\D     Matches any single non-digit
\L     Matches any non-lower-case-letter
\P     Matches any single non-punctuation-mark
\S     Matches any single non-spacing-character
```

\D Matches any non-upper-case-letter
\W Matches any single non-word-character (i.e., matches either \p or \s)

Other character classes

. Matches any single character
[...] Character range: matches any single character inside
[^...] Negated character range: matches any single character not inside

Inside a character range

a-h Any character in the run "e" to "h" inclusive (and so on for other runs)
]... Starting with ">" means that a literal close angle bracket is included
\ Backslash has the same meaning as for backslashed character classes: see above

Structural

| Divides alternatives: "fish|fowl" matches either
(?i) Always matches: switches to case-insensitive matching from here on
(?-i) Always matches: switches to case-sensitive matching from here on

Repetitions

? Matches "..." either 0 or 1 times, i.e., makes "..." optional
* Matches "..." 0 or more times: e.g. "\s*" matches an optional run of space
+ Matches "..." 1 or more times: e.g. "x+" matches any run of "x"s
{6} Matches "..." exactly 6 times (similarly for other numbers, of course)
{2,5} Matches "..." between 2 and 5 times
{3,} Matches "..." 3 or more times
"..?" ?after any repetition makes it "lazy", matching as few repeats as it can

Numbered subexpressions

(...) Groups part of the expression together: matches if the interior matches
$1 Matches the contents of the 1st subexpression reading left to right
$2 Matches the contents of the 2nd, and so on up to "\9" (but no further)

Unnumbered subexpressions

(#...) Comment: always matches, and the contents are ignored
(?=...) Lookahead: matches if the text ahead matches "...", but doesn't consume it
(?!...) Negated lookahead: matches if lookahead fails
(?<=...) Lookbehind: matches if the text behind matches "...", but doesn't consume it
(?<!...) Negated lookbehind: matches if lookbehind fails
(?>...) Possessive: tries to match "..." and if it succeeds, never backtracks on this
(?(1)...) Conditional: if \1 has matched by now, require that "..." be matched
(?(1)...|...) Conditional: ditto, but if \1 has not matched, require the second part
(?(?=...)...) Conditional with lookahead as its condition for which to match
(?(?<=...)...) Conditional with lookbehind as its condition for which to match

IN REPLACEMENT TEXT

\char If char is other than a-z, A-Z, 0-9 or space, expands to that literal char
\\ In particular, "\\" expands to a literal backslash "\"
\n Expands to a line break character
\t Expands to a tab character (but use this only with external files)
$0 Expands to the full text matched
$1 Expands to whatever the 1st bracketed subexpression matched
$2 Expands to whatever the 2nd matched, and so on up to "\9" (but no further)
\l0 Expands to \0 converted to lower case (and so on for "\l1" to "\l9")
\u0 Expands to \0 converted to upper case (and so on for "\u1" to "\u9")

Start of Chapter 20: Advanced Text
Back to §20.8. Replacements
Onward to Chapter 21: Lists: §21.1. Lists and entries

Examples from Chapter 20: Advanced Text

412 Example Capital City *RB*

To arrange that the location information normally given on the left-hand side of the status line appears in block capitals.

Not much is needed for this. The only noteworthy point is that it doesn't work by changing the LHSL to "[the player's surroundings in upper case]": it cannot do this because "the player's surroundings" is not a value. Instead, "[the player's surroundings]" is a text substitution sometimes printing the name of a room, sometimes printing "Darkness", and so on. We must therefore load it into a text first, and then apply "...in upper case".

```
"Capital City"

Capital City is a room. East is Lower Caissons. South of
Lower Caissons
is San Seriffe. East of San Seriffe is a dark room.

To say the player's capitalised surroundings:
    let the masthead be "[the player's surroundings]" in
upper case;
    say the masthead.

When play begins:
    now the left hand status line is "[the player's
capitalised surroundings]".

Test me with "e / s / e".
```

We can now change the case of any text produced by a "to say..." phrase. This is often useful when we would like to make use of a standard say phrase in some new context. Say, for instance, that we would like to "[is-are the list...]" in a context that needs the first letter to be capitalized.

We could write a new say phrase, such as "to say is-are the list of (N - a description of objects) in sentence capitalization"; but there is an easier way, and that is to set a text variable to the output of the to say phrase, and then print that text in the case of our choice.

For example:

```
"Rocket Man"

Instead of going somewhere from the spaceport when
the player carries something:
    let N be "[is-are the list of things carried by the player]
really suitable gear to take to the moon?" in sentence
case;
    say "[N][paragraph break]".

The Spaceport is a room. North of the Spaceport is the
Rocket Launch Pad. The player carries a stuffed bear, a
chocolate cookie, and a book.

The description of the book is "It is entitled [italic
type]Why Not To Take [sentence cased inventory] To
The Moon[roman type]."

To say sentence cased inventory:
    let N be "[a list of things carried by the player]" in title
case;
```

```
    say "[N]".


Test me with "n / x book".
```

ExampleAlpha *RB*

Creating a beta-testing command that matches any line starting with punctuation.

Sometimes we want to let testers of a game insert their own comments during a transcript, without those comments wasting turns of the game or producing lengthy or inappropriate parser errors. Many testers have a habit of prefacing comments with a punctuation mark, so let's say that we'd like to catch any command that starts with any punctuation at all:

```
"Alpha"


When play begins:
    say "Hi, Larry! Thanks for testing my game!!"


Unimplemented Room is a room. "Room description
goes here..."


The scary troll is a man in Unimplemented Room.


After reading a command (this is the ignore beta-
comments rule):
    if the player's command matches the regular
expression "^\p":
        say "(Noted.)";
        reject the player's command.
```

Test me with "x me / x troll / !this game is a bit dull so far / kiss troll / ? does this troll do anything? / :yawn".

---

415 **Example** **About Inform's regular expression support** *RB*

Some footnotes on Inform's regular expressions, and how they compare to those of other programming languages.

There is not really any unanimity about what regular expression language is. The unix tools sed and grep extend on Kleene's original grammar. Henry Spencer's regex library extended on this again, and was a foundation for Perl, but Perl once again went further. Philip Hazel's PCRE, despite the name Perl Compatible Regular Expressions, makes further extensions still, and so on.

Inform's regular expressions are modelled on those of Perl, as the best de facto standard we have, but a few omissions have been inevitable. Inform's regex matcher must occupy source code less than one hundredth the size of PCRE, and it has very little memory. Inform aims to behave exactly like Perl except as follows:

(i) Inform allows angle brackets as synonymous with square brackets, for reasons explained above. This means literal angle brackets have to be escaped as "\<" and "\>" in Inform regular expressions, which is unnecessary in Perl.

(ii) Inform only has single-line mode, not multiline mode: this removes need for the mode-switches "(?m)" and "(?s)" and the positional markers "\A" and "\Z". Multiline mode is idiosyncratic to Perl and is a messy compromise to do with holding long files of text as single strings, yet treating them as lists of lines at the same time: this would not be sensible for Inform. Similarly, because there is no ambiguity about how line breaks are represented in Inform strings (by a single "\n"), initial newline convention markers such as "(*ANYCRLF)" are unsupported.

(iii) The codes "\a", "\r", "\f", "\e", "\0" for alarm, carriage return, form feed, escape and the zero character are unsupported: none of these can occur in an Inform string.

(iv) Inform does not allow characters to be referred to by character code (whereas Perl allows "\036" for an octal character code, "\x7e" for a hexadecimal one, "\cD" for a control character). This is because we do not want the user to know whether text is internally stored as ZSCII or Unicode.

(v) Inform's character class "\p" (and its negation "\P") have no equivalent in Perl, and Inform's understanding of "\w" is different. Perl defines this as an upper or lower case English letter, underscore or digit, which is good for programming-language identifiers, but bad for natural language - for instance, "é" is not matched by "\w" in Perl, but unquestionably it appears in words. Inform therefore defines "\w" as the negation of "\s" union "\p".

(vi) Inform supports only single-digit grouping numbers "\1" to "\9", whereas Perl allows "\10", "\11", ...

(vii) POSIX named character ranges are not supported. These are only abbreviations in any case, and are not very useful. (Note that the POSIX range "[:punct:]", which is supposedly for punctuation, includes many things we do not want to think of that way - percentage signs, for instance - and so "\p" has a more natural-language-based definition.)

(viii) Character classes can be used inside ranges, so that "<\da-f>" is legal, but not as ends of contiguous runs, so that "<\d-X>" is not legal. (As reckless as this is, it is legal in Perl.)

(ix) For obvious reasons, escapes to Perl code using the notation "(?{...})" are unsupported, and so is the Perl iteration operator "\G".

(x) Perl's extended mode "(?x)", a lexical arrangement which allows expressions to be expanded out as little computer programs with comments, is unsupported. It would look awful syntax-coloured in the Inform interface and is not a style of coding to be encouraged.

Inform further does not support the Python extension of named subexpression groups, nor the Java extension of the possessive quantifier "++". There was only so much functionality we could squeeze in.

As verification of Inform's matching algorithm, we took the Perl 5 source code's notorious "re-test.txt" set of 961 test cases, removed the 316 using features unsupported by Inform (220 tested multiline mode, for instance), and ran the remaining 645 cases through Inform. It agrees with Perl on 643 of these: the two outstanding are -

(i) Perl is able to match "^(a\1?){4}$" against "aaaaaa" but Inform is not - Inform's backtracking is not as good when it comes to repetitions of groupings which are recursively defined. (Note that the optional "\1" match refers to the value of the bracketed expression which contains it, so that the interpretation is different on each repetition. Here to match we have to interpret "?" as 0, 0, 1, 0 repeats respectively as we work through the "{4}".)

(ii) Perl matches "((<a-c>)b*?\2)*" against "ababbbcbc" finding the match "ababb", whereas Inform finds the match "ababbbcbc". This is really a difference of opinion about whether the outer asterisk, which is greedy, should be allowed three matches rather than two if to do so requires the inner asterisk, which is not greedy, to eat more than it needs on one of those three matches.

Case (i) is a sacrifice to enable Inform's back-tracking to use less memory. Case (ii) simply seems unimportant.

Example **Identity Theft** *RB*

Allowing the player to enter a name to be used for the player character during the game.

Let's say we want to allow the player to enter any name he likes for his character. Moreover, we want to reject very long names (which are likely to be mistakes anyway), and we want to extract the player's chosen first name from the rest.

```
"Identity Theft"


The player's forename is a text that varies. The player's
full name is a text that varies.


When play begins:
    now the command prompt is "What is your name? >
".


To decide whether collecting names:
    if the command prompt is "What is your name? > ",
yes;
    no.


After reading a command when collecting names:
    if the number of words in the player's command is
greater than 5:
        say "[paragraph break]Who are you, a member of
the British royal family? No one has that many names.
Let's try this again.";
        reject the player's command;
    now the player's full name is the player's command;
    now the player's forename is word number 1 in the
player's command;
    now the command prompt is ">";
    say "Hi, [player's forename]![paragraph break]";
```

```
    say "[banner text]";
    move the player to the location;
    reject the player's command.
```

We also want to postpone the proper beginning of the game until we've gotten the name:

```
Instead of looking when collecting names: do nothing.

Rule for printing the banner text when collecting names:
do nothing.

Rule for constructing the status line when collecting
names: do nothing.

Your Bedroom is a room. The printed name of Your
Bedroom is "[player's forename]'s Bedroom".

The player carries a letter. The description of the letter
is "Dear [player's full name], [paragraph break]You have
won the Norwegian Daily Lottery! ...".
```

If we are compiling for Glulx, this is enough to capture not only the player's name but also the capitalization he uses.

If we are compiling for the Z-machine, the player's input will unfortunately be reduced to lower case before we can inspect it. If we would like by default to capitalize the first letter of each word of the name, we might substitute the following after reading a command rule:

```
After reading a command when collecting names:
    if the number of words in the player's command is
greater than 5:
        say "[paragraph break]Who are you, a member of
the British royal family? No one has that many names.
Let's try this again.";
```

```
        reject the player's command;
    now the player's full name is the substituted form of
"[the player's command in title case]";
    now the player's forename is word number 1 in the
player's full name;
    now the command prompt is ">";
    say "Hi, [player's forename]![paragraph break]";
    say "[banner text]";
    move the player to the location;
    reject the player's command.
```

---

417 **Example** **Mirror, Mirror** *RB*

The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

```
"Mirror, Mirror"

The Sorcerer's Workshop is a room. "The sorcerer's den
is a dusty, whispering place. A grandfather clock with
skeletal hands reads [the time of day in words]. The
floorboards are stained where that porridge just wouldn't
come out."

The Apprentice's Pantry is east of the Workshop. "This
is where the aproned apprentice traditionally makes the
camomile tea, cleans out the jackdaw cages and
furtively examines purloined artefacts."

When play begins: erase the mirror.

The player carries a magic mirror. The magic mirror has
a text called the mirror vision.
```

To erase the mirror: now mirror vision of the mirror is "The mirror is polished clean, and has no impression upon it."

To say current room description: try looking.
To expose the mirror:
    say "The mirror shines momentarily with a dazzling light.[paragraph break]";
    now mirror vision of the mirror is the substituted form of "The hazy image in the mirror preserves a past sight:[line break][current room description]All is distorted and yet living, as though the past and present are coterminous in the mirror."

Understand "hold up [something preferably held]" or "hold [something preferably held] up" as holding aloft.
Holding aloft is an action applying to one carried thing.
Report holding aloft: say "You hold [the noun] aloft."

Instead of rubbing the mirror: erase the mirror; try examining the mirror. Instead of holding aloft the mirror: expose the mirror.

The description of the mirror is "[mirror vision of the mirror]".

Test me with "look / examine mirror / hold up mirror / z / look / x mirror / rub mirror / east / hold mirror up / west / x mirror".

when several matches go out at once.

Here we create a class of matches that can be used to burn other objects. Our objectives are as follow:

Burned objects other than matches should be removed from play instantly (just as edible objects are instantly eaten). We could give everything its own burning duration, but that complicates matters and allows for fire to spread from one object to another; for an example of how to do that, see the example "In Fire or in Flood".

Matches should be described to show whether they are burning or extinguished, and when the parser chooses one of several identical matches, it should make very clear which match it has selected.

The game must sensibly select and, if necessary, automatically light new matches to carry out a >BURN THING command.

The matches must burn for a set number of turns before going out, never to be used again.

And finally, the part for which the text will be useful: when several matches go out in the same turn, we want the game to print

```
Four matches go out.
```

rather than

```
A match goes out.
A match goes out.
A match goes out.
A match goes out.
```

This last function appears down in Section 3, if we wish to skip ahead and look at it.

```
"The Cow Exonerated"

Section 1 - Simple Burning
```

Understand the commands "light" and "burn" as something new.

Understand "burn [something] with [strikable-match]" as burning it with. Understand "burn [something] with [something preferably held]" as burning it with. Burning it with is an action applying to one thing and one carried thing.

Understand the command "light" as "burn".

A thing can be flammable or impervious. A thing is usually impervious.

Check burning something with something (this is the burn only with flaming matches rule):
    if the second noun is not a strikable-match, say "You can only light things with matches." instead;
    if the second noun is not flaming, say "[The second noun] needs to be burning first." instead.

Check burning something with something (this is the burn only flammable things rule):
    if the noun is impervious, say "[The noun] cannot be burned." instead.

Check burning something with something (this is the burn only things not held rule):
    say "[one of]It occurs to you to set down [the noun] before burning, just for safety's sake. [or]Again, you decide to put down [the noun] prior to burning. [or]You try setting down [the noun] as usual. [stopping][run paragraph on]";
    silently try the player dropping the noun;

```
    if the player encloses the noun, stop the action.

Carry out burning something with something (this is the
simplistic burning rule):
    now the noun is nowhere.

Report burning something with something:
    say "You burn up [the noun]."

Rule for implicitly taking the second noun while burning
something with something which is not a strikable-
match:
    say "You can only light things with matches.";
    stop the action.
```

Section 2 - Matches

The word "matches" is used by Inform to compare snippets of text (see "Reading a command" in the Activities chapter). This can sometimes cause awkwardness if we also have a kind called "match", so for the occasion we will give our matches a more specialized name, never visible to the player:

```
A strikable-match is a kind of thing. The plural of
strikable-match is s-matches.

A strikable-match has a number called duration. The
duration of a strikable-match is usually 3.

Rule for printing the name of a strikable-match: say
"match".
Rule for printing the plural name of a strikable-match:
say "matches".
```

```
Understand "match" as a strikable-match. Understand
"matches" as a strikable-match.


Flame-state is a kind of value. The flame-states are
burnt, flaming, and new. Understand "burning" or "lit" as
flaming. Understand "unused" as new.


A strikable-match has a flame-state. A strikable-match is
usually new. Understand the flame-state property as
describing a strikable-match.


Before printing the name of a strikable-match while
asking which do you mean:
    say "[flame-state] ".


Before printing the name of a strikable-match while
taking inventory:
    say "[flame-state] ".
Before printing the plural name of a strikable-match
while taking inventory:
    say "[flame-state] ".


Before printing the name of a strikable-match while
clarifying the parser's choice of something:
    if not taking inventory, say "[flame-state] ".


After printing the name of a strikable-match (called
special-target) while clarifying the parser's choice of
something:
    if the player carries the special-target:
        say " you're carrying";
    otherwise if the special-target is in the location:
        say " on the ground";
    otherwise:
```

```
        say " [if the holder of the special-target is a
container]in[otherwise]on[end if] [the holder of the
special-target]".


Understand "strike [something]" as attacking.


Understand "strike [strikable-match]" as striking. Striking
is an action applying to one carried thing.


Understand "burn [strikable-match]" as striking.


Does the player mean striking a new strikable-match:
    it is very likely.


Does the player mean striking a burnt strikable-match:
    it is unlikely.


Check striking a strikable-match (this is the strike only
new matches rule):
    if the noun is burnt, say "[The noun] has already burnt
down and cannot be relit." instead;
    if the noun is flaming, say "[The noun] is already
burning." instead.


Carry out striking a strikable-match (this is the standard
striking rule):
    now the noun is flaming;
    now the noun is lit.


Report striking a strikable-match (this is the standard
report striking rule):
    say "You light [the noun]."
```

```
Before burning something with a new strikable-match
(this is the prior lighting rule):
    say "(first [if the player does not carry the second
noun]taking and [end if]lighting [the second noun])
[command clarification break]";
    silently try striking the second noun;
    if the second noun is not flaming, stop the action.


Rule for implicitly taking a strikable-match (called target)
while striking:
    try silently taking the target.


Does the player mean burning something with a flaming
strikable-match:
    it is very likely.


Does the player mean burning something with a new
strikable-match:
    it is likely.


Does the player mean burning something with a burnt
strikable-match:
    it is unlikely.


Instead of burning a burnt strikable-match with
something:
    say "[The noun] is completely consumed and cannot
be relit."


Section 3 - Putting the Matches Out


Every turn:
```

```
    let N be 0; [here we track how many matches are
being put out during this turn, so that we don't have to
mention each match individually if several go out during
the same move]
    repeat with item running through flaming s-matches:
        decrement the duration of the item;
        if the duration of the item is 0:
            now the item is burnt;
            now the item is unlit;
            if the item is visible, increment N;
    if N is 1:
        say "[if the number of visible flaming s-matches is
greater than 0]One of the matches [otherwise if the
number of burnt visible s-matches is greater than 1]Your
last burning match [otherwise]The match [end if]goes
out.";
    otherwise if N is greater than 1:
        let enumeration be "[N in words]";
        if N is the number of visible s-matches:
            if N is two, say "Both";
            otherwise say "All [enumeration]";
        otherwise:
            say "[enumeration in title case]";
        say " matches go out[if a visible strikable-match is
flaming], leaving [number of visible flaming s-matches in
words] still lit[end if]."
```

Section 4 - Scenario

```
Old Chicago is a room.

The player carries a flammable thing called a log.
Understand "wooden" and "wood" as the log.

The player carries two s-matches. The matchbox is an
open openable container. It contains five s-matches.
The player carries the matchbox.

When play begins:
```

```
    now every strikable-match carried by the player is
flaming;
    now every strikable-match carried by the player is lit.


Test me with "i / burn match / i / i / burn log with match /
burn matchbox with match / i".
```

4.19 ExampleBlackout *RB*

Filtering the names of rooms printed while in darkness.

In this example, we want the names of rooms to be asterisked out if the player wanders around without the benefit of a candle. We can do this by treating the room names as text, then replacing every letter:

```
"Blackout"


Tiny Room is a dark room. Absurdly Long-Named Room
is a dark room. It is west of Tiny Room.


The Candle Factory is north of Tiny Room. It contains a
beeswax candle. The beeswax candle is lit.


Rule for printing the name of a dark room:
    let N be "[location]";
    replace the regular expression "\w" in N with "*";
    say "[N]".


Test me with "w / look / e / n / get candle / s / w".
```

Notice that the hyphen in the Absurdly Long-Named Room does not get replaced. We could replace even that, if we liked, with

```
replace the regular expression "\S" in N with "*";
```

which would catch every character that is not a space.

---

420 **ExampleFido** *RB*

A dog the player can name and un-named at will.

Suppose we'd like to have a dog which the player is allowed to name himself. We'd like to deal correctly with both

```
>name the dog fido
```

and

```
>name the dog "fido"
```

so we'll also need to strip quotation marks out of the command. We can do this as follows:

```
"Fido"

The Back Yard is a room.

A dog is an animal in Back Yard. The dog has some text
called the nickname. The nickname of the dog is
"nothing". Understand the nickname property as
describing the dog.

Rule for printing the name of the dog when the
```

```
nickname of the dog is not "nothing":
    say "[nickname of the dog]"



Naming it with is an action applying to one thing and
one topic. Understand "name [something] [text]" as
naming it with. Check naming it with: say "You can't
name that."



Instead of naming the dog with "nothing":
    now the nickname of the dog is "nothing";
    now the dog is improper-named;
    say "You revoke your choice of dog-name."



Instead of naming the dog with something:
    let N be "[the topic understood]";
    replace the text "'" in N with "";
    now the nickname of the dog is "[N]";
    now the dog is proper-named;
    say "The dog is now known as [nickname of the
dog]."



Test me with "name the dog Fido / name the dog
Lawrence / look / x lawrence / name Lawrence nothing /
look / x lawrence".
```

---

421 **Example Igpay Atinlay** *RB*

A pig Latin filter for the player's commands.

For the sake of argument, suppose we want to parrot back all the player's commands in pig Latin:

```
"Igpay Atinlay"
```

```
Armfay is a room.


After reading a command:
    let N be "[the player's command]";
    replace the regular expression "\b(<aeiou>+)(\w*)" in
N with "\1\2ay";
    replace the regular expression
"\b(<bcdfghjklmnpqrstvwxz>+)(\w*)" in N with "\2\1ay";
    say "[N][paragraph break]";
    reject the player's command.


Test me with "nix on the stupid".
```

---

422 **Example** **Mr. Burns' Repast** *RB*

Letting the player guess types for an unidentifiable fish.

Suppose we have an unhappily mutated fish that the player can refer to by any of a number of species names, or any word followed by -fish. We want to reject these commands, but preserve a memory of what the player last tried to call the thing:

```
"Mr. Burns' Repast"


Wharf is a room.


There is an unknown fish in the Wharf. The unknown
fish has some a text called the supposed name. The
description of the unknown fish is "The victim of heavy
mutagens, this thing is not really recognizable as any
species you know.".
```

```
Fish variety is a kind of value. The fish varieties are
salmon, albacore, mackerel.


Rule for printing the name of the unknown fish:
    if the supposed name of the unknown fish is "", say
the printed name of the unknown fish;
    otherwise say the supposed name of the unknown
fish.


After reading a command:
    if the unknown fish is visible and player's command
matches the regular expression "\b\w+fish":
        let N be "[the player's command]";
        replace the regular expression ".*(?=\b\w+fish)" in
N with "";
        now N is "[N](?)";
        now the supposed name of the unknown fish is N;
        respond with doubt;
        reject the player's command;
    otherwise if the unknown fish is visible and the
player's command includes "[fish variety]":
        now supposed name of the fish is "[fish variety
understood](?)";
        respond with doubt;
        reject the player's command.


To respond with doubt:
    say "You're not [italic type]sure[roman type] you're
seeing any such thing."


Test me with "get swordfish / look / touch monkfish /
look / listen to tunafish / x fish / x salmon / look".
```

ExampleNorthstar
Making Inform understand ASK JOSH TO TAKE INVENTORY as JOSH, TAKE

INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

Most of the time, Inform understands commands to other characters when they take the form "JOSH, TAKE INVENTORY" or "JOAN, WEAR THE ARMOR". But novice players might also try commands of the form ASK JOSH TO TAKE INVENTORY or ORDER JOAN TO WEAR THE ARMOR.

The easiest way to make Inform understand such commands is to meddle directly with the player's command, changing it into the format that the game will understand, as here:

```
"Northstar"

The Northstar Cafe is a room. "The Northstar is
crammed with its usual brunch crowd, and you were
lucky to get a table at all. You are now awaiting the
arrival of your ricotta pancakes."

Josh is a man in The Northstar Cafe. "Josh is on his
way past your table." The description of Josh is "He is a
waiter here, but you also know him socially, so he tends
to be more chatty than the other waiters." A persuasion
rule: persuasion succeeds.

After reading a command:
    let N be "[the player's command]";
    replace the regular expression "\b(ask|tell|order) (.+?)
to (.+)" in N with "\2, \3";
    change the text of the player's command to N.

Test me with "ask Josh to take inventory / tell Josh to
take inventory / order Josh to take inventory".
```

Note that we have to copy N back explicitly to replace the player's command.

Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

Novice players of interactive fiction, unfamiliar with its conventions, will often try to add extra phrases to a command that the game cannot properly parse: HIT DOOR WITH FIST, for instance, instead of HIT DOOR.

While we can deal with some of these instances by expanding our range of actions, at some point it becomes impossible to account for all the possible prepositional phrases that the player might want to tack on. So what do we do if we want to handle those appended bits of text sensibly?

We could go through and remove any piece of text containing "with ..." from the end of a player's command; the problem with that is that it overzealously lops off the ends of valid commands like UNLOCK DOOR WITH KEY, as well. So clearly we don't want to do this as part of the "After reading a command..." stage.

A better time to cut off the offending text is right before issuing a parser error. At that point, Inform has already determined that it definitely cannot parse the instruction as given, so we know that there's something wrong with it.

The next problem, though, is that after we've edited the player's text we want to feed the corrected version back to Inform and try once more to interpret it.

This is where we have a valid reason to write a new "rule for reading a command". We will tell Inform that when we have just corrected the player's input to something new, it should not ask for a new command (by printing a prompt and waiting for another line of input); it should instead paste our stored corrected command back into "the player's command" and proceed as though that new text had just been typed.

Thanks to John Clemens for the specifics of the implementation.

"Cave-troll" by JDC

Section 1 - The Mechanism

The last command is a text that varies.

The parser error flag is a truth state that varies. The parser error flag is false.

Rule for printing a parser error when the latest parser error is the only understood as far as error and the player's command matches the text "with":
    now the last command is the player's command;
    now the parser error flag is true;
    let n be "[the player's command]";
    replace the regular expression ".* with (.*)" in n with "with \1";
    say "(ignoring the unnecessary words '[n]')[line break]";
    replace the regular expression "with .*" in the last command with "".

Rule for reading a command when the parser error flag is true:
    now the parser error flag is false;
    change the text of the player's command to the last command.

Section 2 - The Scenario

The Cave is a room.

The troll is a man in the cave.

The player carries a sword.

```
The chest is a locked lockable container in the cave.


Test me with "attack troll with sword / unlock chest with
sword / attack troll as a test".
```

A caveat about using this method in a larger game: "parser error flag" will not automatically control the behavior of any rules we might have written for Before reading a command... or After reading a command..., so they may now fire at inappropriate times. It is a good idea to check for parser error flag in those rules as well.

Chapter 21: Lists

ts of objects; *§21.7. Lists of values matching a description*; *§21.8. Sorting, reversing and rotating lists*; §

Contents of *Writing with Inform*
Chapter 20: Advanced Text
Chapter 22: Advanced Phrases
Indexes of the examples

§21.1. Lists and entries

Many sections in this book begin by introducing a new kind of value. Reading through in order, the possibilities mount up: numbers, times, texts, and so on. (See the Kinds page of the Index for a convenient list of the options.) This section is a little different: rather than showing a single new kind of value, it shows how to make a new kind out of any existing one.

If K is any kind of value, then "list of K" is also a kind of value. For instance, we could write:

```
let L be a list of numbers;
```

and this would create a new "let" variable, called L, whose kind of value is "list of numbers". On the other hand, we are not allowed to write:

```
let L be a list;
```

because "list" by itself is not a kind of value. (Inform always needs to know what kinds the values entered in a list are going to have.)

Lists are like flexible-length table columns, but that probably makes them sound more mysterious than they really are. A list is simply a sequence of values, called its "entries", numbered from 1 upwards. The number of entries is called its "length". If we try

```
let L be a list of numbers;
```

```
say "L has [the number of entries in L] entries.";
```

then we find

```
L has 0 entries.
```

This is because all lists start out empty when created: that is, they initially have 0 entries. Inform has two built-in adjectives "empty" and "non-empty" which can apply to lists, and they mean just what they ought to mean: a list is empty if its length is 0, and otherwise non-empty.

We can add entries very easily:

```
add 2 to L; add 3 to L; add 5 to L;
```

We can now, for instance, try saying the list:

```
say "L is now [L].";
```

with the result

```
L is now 2, 3 and 5.
```

Note that only numbers can be added to L: if we try

```
add "clock" to L;
```

Inform will produce a problem message, because L has kind "list of numbers", whereas "clock" is text. In this way, Inform ensures that a list always contains values of the same kind throughout. So it's not possible to construct a list whose entries are:

```
2, "fish", 4 and the Entire Game
```

Such a list would be very hazardous to deal with, in any case. If what we need is a combination of different kinds of values, tables are a better option.

Finally, note that since "list of numbers" is a kind of value in its own right, so is "list of

lists of numbers", and so on - though such lists are trickier to deal with, they are sometimes handy.

Start of Chapter 21: Lists

Back to Chapter 20: Advanced Text: §20.9. Summary of regular expression notation

Onward to §21.2. Constant lists

§21.2. Constant lists

It is convenient to have a concise way to write down a constant list. Just as we could write "231", say, or "7:01 AM" to refer to particular number and time constants, so we can write list constants:

```
let L be {1, 2, 3, 4};
```

Inform recognises that "{1, 2, 3, 4}" is a list because of the braces, and looks at the entries inside, sees that they are numbers, and deduces that it is a constant whose kind of value is "list of numbers". L is then a temporary list variable and we can add to it, remove things, and so on as we please - {1, 2, 3, 4} is merely its initial value.

When constructing lists, it is worth noting that Inform requires spaces after the commas (which seems a little harsh, but is necessary because otherwise many sensible literal specifications for units would be impossible - anyway, the reason isn't important here). So

```
let L be {1,2,3,4};
```

would produce problem messages. But Inform does not require spaces round its braces.

We call this way of writing a list "brace notation". In mathematics, braces are usually used for sets, and properly speaking these are sequences not sets - so that "{1, 2, 3, 4}" is different from "{4, 3, 2, 1}" - but it is still a familiar notation. Similarly,

```
let L be {"apple", "pear", "loganberry"};
```

makes L a list of texts; and

```
The marshmallow, the firework and the stink bomb are in the Scout
Hut. The list of prohibited items is a list of objects that varies. The list
of prohibited items is {the firework, the stink bomb}.
```

makes a global variable ("list of prohibited items") with kind of value "list of objects", and whose initial value is to contain two things: the firework and the stink bomb. More exotically, if we need to make lists of lists:

```
let L be {{1, 2}, {6, 7, 8}};
```

gives L the kind of value "list of lists of numbers", with (initially) two entries: the list {1, 2} (a list of numbers), then the list {6, 7, 8} (ditto).

Constant lists are convenient, too, when a column in a table needs to contain lists:

```
The duck, the orange, the cider, the cinnamon and the orange are in
the Kitchen.

Table of Requirements

recipe                  ingredients
"duck a l'orange"       {the duck, the orange}
"spiced cider"          {the cider, the cinnamon, the orange}
```

A special word about the constant list "{ }". This means the list with no entries - the empty list. If we try to create a new "let" variable M with

```
let M be { };
```

then Inform will produce a problem message, because it cannot tell what sort of list M will be: a list of numbers, or texts, or times, or...? On the other hand, writing

```
let M be { };
```

is fine provided that M already exists, and then does the obvious thing - empties M. Similarly, a table column in which every entry is "{ }" produces a problem message unless the heading for that column spells out the kind of value stored within it: for instance, "ingredients (list of texts)".

All of this is a notation for constant lists only, not some sort of gluing-things-together operation. So this, for instance:

```
let L be {100, the turn count};
```

is not allowed, even though "the turn count" is a number: because it is a number that varies, the braces do not contain constants, and therefore this is not a list constant.

Start of Chapter 21: Lists
Back to §21.1. Lists and entries
Onward to §21.3. Saying lists of values

§21.3. Saying lists of values

Any list L can be said, provided that its contents can be said. For example:

```
let L1 be {2, 3, 5, 7, 11};
say L1;
```

produces the text "2, 3, 5, 7 and 11" - unless we have "Use serial comma." set, in which case a comma appears after the 7. We also have the option of using the more formal notation:

say "[(list of values) **in brace notation]"**

This text substitution produces the list in the form of "{", then a comma-separated list, and then "}", which looks less like an English sentence but more mathematical. Example:

```
"[list of people in brace notation]"
```

might produce "{ yourself, Mr Darcy, Flashman }".

If we say a list of lists, then the individual entry lists are always printed in brace notation: the ordinary sentence way would be incomprehensible.

Of course, the values in L1 are written out in number form because L1 is a list of numbers: we could alternatively try

```
let L2 be {the piano, the music stand};
say L2;
```

which produces "piano and music stand". Lists of objects can be said in two additional ways:

say "[(list of objects) **with definite articles]"**

This text substitution writes out the list in sentence form, adding the appropriate definite articles. Example:

```
let L be {the piano, the music stand};
say "[L with definite articles]";
```

says "the piano and the music stand".

say "[(list of objects) **with indefinite articles]"**

This text substitution writes out the list in sentence form, adding the appropriate indefinite articles. Example:

```
let L be {the piano, the music stand};
say "[L with definite articles]";
```

says "a piano and a music stand".

Start of Chapter 21: Lists
Back to §21.2. Constant lists

Onward to §21.4. Testing and iterating over lists

Example 425: ★ **Oyster Wide Shut** Replacing Inform's default printing of properties such as "(closed)", "(open and providing light)", etc., with our own, more flexible variation.

---

§21.4. Testing and iterating over lists

If L is a list, we can interrogate it to see whether it does or does not contain (at least one instance of) any compatible value V:

if (value) **is listed in** (list of values)**:**

This condition is true if the given value, which must be of a compatible kind, is one of those in the list. For instance, if L is our list of the numbers 2, 3, 5, 7 and 11 then 5 is listed in it but 6 is not.

if (value) **is not listed in** (list of values)**:**

This condition is true if the given value, which must be of a compatible kind, is not one of those in the list.

We can also repeat running through a list (just as we can with table rows). Thus:

repeat with (a name not so far used) **running through** (list of values)**:**

This phrase causes the block of phrases following it to be repeated once for each item in the given list, storing that value in the named variable. (The variable exists only temporarily, within the repetition.) Example:

```
let L be {2, 3, 5, 7, 11, 13, 17, 19};
repeat with prime running through L:
```

```
...
```

If the list is empty, nothing happens: the "..." phrase(s) are never tried.

In the next sections, we shall see that it is possible to change, reorder and resize lists. But it's important never to change a list that's being repeated through. The following:

```
let L1 be {1, 2, 3, 4};
repeat with n running through L1:
    remove n from L1;
```

leaves L1 containing {2, 4}, since the removals from the list cause it to shuffle back even while we repeat through it - a bad, bad idea.

Start of Chapter 21: Lists
Back to §21.3. Saying lists of values
Onward to §21.5. Building lists

§21.5. Building lists

We have already seen "add... to...". This in fact comes in two forms:

add (value) **to** (list of values)

This phrase adds the given value to the end of the list. Example:

```
let L be {60, 168};
add 360 to L;
```

results in L being {60, 168, 360}. Note that the value is added even if it already occurs somewhere in L; this can be avoided with "if absent". So:

```
add 168 to L, if absent;
```

would do nothing - it is already there.

add (list of values) **to** (list of values)

This phrase adds the first list to the end of the second. Example:

```
let L be {2, 3, 5, 7};
add {11, 13, 17, 19} to L;
```

results in L being {2, 3, 5, 7, 11, 13, 17, 19}.

If we don't want to add new entries at the end, we can instead say where they should go:

add (value) **at entry** (number) **in** (list of values)

This phrase adds the given value so that it becomes the entry with that index number in the list. Example:

```
let L be {1, 2, 3, 4, 8, 24};
add 12 at entry 6 in L;
```

sets L to {1, 2, 3, 4, 8, 12, 24}. If there are N entries in L, then we can add at any of entries 1 up to N+1: adding at entry N+1 means adding at the end. The phrase option "if absent" makes the phrase do nothing if the value already exists anywhere in L.

add (list of values) **at entry** (number) **in** (list of values)

This phrase adds the first list to the second so that it begins at the given position. Example:

```
let L be {1, 2, 3, 4};
add {4, 8, 12} at entry 3 in L;
```

results in L being {1, 2, 4, 8, 12, 3, 4}.

A list is allowed to contain duplicates, and the order matters. For instance:

```
let L be {2, 2, 3};
```

makes L into "2, 2 and 3". This is a different list to the one made by:

```
let M be {2, 3, 2};
```

even though L and M have the same values, repeated the same number of times - for two lists to be equal, they must have the same kind of entry, the same number of entries, and the same entries in each position.

We can also strike out values:

remove (value) **from** (list of values)

This phrase removes every instance of the given value from the list. Example:

```
let L be {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
remove 1 from L;
```

results in L being {3, 4, 5, 9, 2, 6, 5, 3}. Ordinarily "remove 7 from L" would produce a run-time problem, since L does not contain the value 7, but using the "if present" option lets us off this: the phrase then does nothing if L does not contain the value to be removed.

remove (list of values) **from** (list of values)

This phrase removes every instance of any value in the first list from the second. Example:

```
let L be {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
remove {0, 2, 4, 6, 8} from L;
```

results in L being {3, 1, 1, 5, 9, 5, 3}. If both lists are large, this can be a slow process, and we might do better by sorting them and trying a more sophisticated method. But this is convenient for anything reasonable-sized.

Again, we can also remove from specific positions:

remove entry (number) **from** (list of values)

This phrase removes the entry at the given position, counting from 1 as the first entry. (Once it is removed, the other entries shuffle down.) Example:

```
let L be {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
remove entry 3 from L;
```

results in L being {3, 1, 1, 5, 9, 2, 6, 5, 3}.

remove entries (number) **to** (number) **from** (list of values)

This phrase removes the entries at the given range of positions, counting from 1 as the first entry. (Once they are removed, the other entries shuffle down.) Example:

```
let L be {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
remove entries 3 to 6 from L;
```

results in L being {3, 1, 2, 6, 5, 3}.

Start of Chapter 21: Lists

Back to §21.4. Testing and iterating over lists

Onward to §21.6. Lists of objects

Example 426: ★ **Robo 1** A robot which watches and records the player's actions, then tries to repeat them back in the same

order when he is switched into play-back mode.

§21.6. Lists of objects

Lists can be made of values of any kind (including other lists), but lists of objects are especially useful. We could always make these "by hand":

```
let L be {the pot plant, the foxglove};
```

But it is usually easier and clearer to use descriptions.

list of (description of values) **... value**

This phrase produces the list of all values matching the given description. Inform will issue a problem message if the result would be an infinite list, or one which is impractical to test: for instance "list of even numbers" is not feasible.

While that works nicely for many kinds of value ("list of recurring scenes", say), it's particularly useful for objects:

```
let L be the list of open containers;
add the list of open doors to L;
```

means that L now contains the open containers (if any) followed by the open doors (if any). Or, for example:

```
let L be the list of things;
remove the list of backdrops from L;
```

makes a list of all non-backdrops.

As mentioned above, lists of objects can be said in two additional ways:

```
"[L with definite articles]"
```

```
"[L with indefinite articles]"
```

And as mentioned below, they can be sorted in property value order:

```
sort L in P order;
sort L in reverse P order;
```

where P is any value property. In all other respects, lists of objects are no different to other lists.

Start of Chapter 21: Lists

Back to §21.5. Building lists

Onward to §21.7. Lists of values matching a description

Example 427: ★ **What Makes You Tick** Building a fishing pole from several component parts that the player might put together in any order.

Example 428: ★★ **Formicidae** Manipulating the order in which items are handled after TAKE ALL.

---

§21.7. Lists of values matching a description

The useful "list of ..." syntax can also be used to produce lists of the values matching a description, too. Thus:

```
let L be the list of non-recurring scenes;
let C be the list of colours;
```

There is little to say here except for the usual warning that some kinds of value have a range which is too large to make this possible. For instance, Inform could not sensibly represent:

```
let N be the list of even numbers;
```

It would just be too large to hold. In general, if we can repeat through, or find the number of, values matching a description, then we can also use "list of" to bring them all together. See the chart of kinds of value in the Kinds index for a project for which kinds of value allow this.

Start of Chapter 21: Lists
Back to §21.6. Lists of objects
Onward to §21.8. Sorting, reversing and rotating lists

§21.8. Sorting, reversing and rotating lists

Any list L can be reversed:

reverse (list of values)

This phrase puts the list in reverse order. The old entry 1 becomes the new last entry, and so on: reversing an empty list or a list containing only one entry leaves it unchanged. Example:

```
let L be {11, 12, 14, 15, 16, 17};
reverse L;
```

results in L being {17, 16, 15, 14, 12, 11}.

And any list can similarly be sorted:

sort (list of values)

This phrase puts the list into ascending order. Example:

```
let L be {6 PM, 11:13 AM, 4:21 PM, 9:01 AM};
sort L;
```

results in L being {9:01 AM, 11:13 AM, 4:21 PM, 6 PM}.

sort (list of values) **in reverse order**

This phrase puts the list into descending order. Example:

```
let L be {6 PM, 11:13 AM, 4:21 PM, 9:01 AM};
sort L in reverse order;
```

results in L being {6 PM, 4:21 PM, 11:13 AM, 9:01 AM}.

sort (list of values) **in random order**

This phrase puts the list into a uniformly random order, shuffling it as if it were a pack of cards. Example:

```
let L be {1, 2, 3, 4, 5, 6};
sort L in random order;
```

might result in L being {3, 1, 5, 6, 4, 2}. Or any of 719 other arrangements, including being left as it was.

Lists of objects can also be sorted in property value order. For instance,

sort (list of objects) **in** (property) **order**

This phrase puts the list into ascending order of the values of the given property for the items in the list; this is only allowed if all of those values do have the property in question. Example:

```
let L be the list of people;
sort L in carrying capacity order;
```

would arrange people with weaklings first, titans last.

sort (list of objects) **in reverse** (property) **order**

This phrase puts the list into descending order of the values of the given property for the items in the list; this is only allowed if all of those values do have the property in question. Example:

```
let L be the list of people;
sort L in reverse carrying capacity order;
```

would arrange people with titans first, weaklings last.

Rotating a list means moving all of its entries along by one place, and then moving the one on the end back to the start. For instance, if L is {1, 2, 3, 4}, then

rotate (list of values)

This phrase shuffles the entries of the list forwards (to the right) by one place, so that the 1st becomes 2nd, the 2nd becomes 3rd, and so on until the last, which becomes the new first entry. Example:

```
let L be { "cow", "heifer", "bullock" };
rotate L;
```

results in L being { "bullock", "cow", "heifer" }.

rotate (list of values) **backwards**

This phrase shuffles the entries of the list backwards (to the left) by one place, so that the 3rd becomes 2nd, the 2nd becomes 1st, and so on; the previous 1st entry becomes the new last entry. Example:

```
let L be { "cow", "heifer", "bullock" };
rotate L backwards;
```

results in L being { "heifer", "bullock", "cow"}. (This achieves the same effect as "reverse L; rotate L; reverse L;" but is a little faster, and a lot less effort to read.)

§21.9. Accessing entries in a list

The length of a list can change as values are added or removed, and can in principle be any number from 0 upwards. A list with 0 entries is empty. We can find the length with:

number of entries in/of (list of values) **... number**

This phrase produces the number of positions in the list. Example:

```
the number of entries in {1, 1, 1, 3, 1}
```

is 5, even though there are only two genuinely different items in the list.

If the length is N then the entries are numbered from 1 (the front) to N (the back). These entries can be accessed directly by their numbers. For instance,

```
entry 2 of L
```

refers to the second entry of L: it can be used as a value, or changed, just as if it were a named variable. For instance, we could write:

```
now entry 7 of L is "Spain";
say "The rain in [entry 7 of L] stays mainly in the plain.";
```

which would (untruthfully) print "The rain in Spain stays mainly in the plain", but only if L had an entry 7 to make use of: if L were a list of 5 entries, say, then a run-time problem results. (And if L cannot hold text, a problem message means that we never get as far as run-time.) Because entries number from 1, this is always incorrect:

```
entry 0 of L
```

and if L is currently empty, then there is no entry which can be accessed, so that any use of "entry ... of L" would produce a run-time problem. There are programming languages in the world where accessing entry 100 in a 7-entry list automatically extends it to be 100 entries long: Inform is not one of them. But see the next section for how to change list lengths explicitly.

Start of Chapter 21: Lists

Back to §21.8. Sorting, reversing and rotating lists

Onward to §21.10. Lengthening or shortening a list

Example 429: ★★★ **Robo 2**
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

---

§21.10. Lengthening or shortening a list

We can explicitly change the length of a list like so:

change (list of values) **to have** (number) **entries**/entry

This phrase alters the given list so that it now has exactly the number of entries given. Example:

```
change L to have 21 entries;
```

If L previously had more than 21 entries, they are thrown away (and lost forever); if L previously had fewer, then new entries are created, using the default value for whatever kind of value L holds. So extending a list of numbers will pad it out with 0s, but extending a list of texts will pad it out with the empty text "", and so on.

We can also write the equivalent phrases:

truncate (list of values) **to** (number) **entries**/entry

This phrase alters the given list so that it now has no more than the number of entries given. Example:

```
truncate L to 8 entries;
```

shortens L to length 8 if it is currently longer than that, trimming entries from the end, but would (for instance) leave a list of length 3 unchanged. Note that

```
truncate L to 0 entries;
```

empties it to { }, the list with nothing in.

truncate (list of values) **to the first** (number) **entries**/entry

This phrase alters the given list so that it now consists only of the initial part of the list with the given length. Example:

```
truncate L to the first 4 entries;
```

turns {1, 3, 5, 7, 9, 11} to {1, 3, 5, 7}.

truncate (list of values) **to the last** (number) **entries**/entry

This phrase alters the given list so that it now consists only of the final part of the list with the given length. Example:

```
truncate L to the last 4 entries;
```

turns {1, 3, 5, 7, 9, 11} to {5, 7, 9, 11}.

But we don't have to truncate: we can also -

extend (list of values) **to** (number) **entries**/entry

This phrase pads out the list with default values as needed so that it now has at least the given length. (If the list is already at least that length, nothing is done.) Example:

```
extend L to 80 entries;
```

lengthens L to length 80 if it is currently shorter than that.

For example,

```
To check sorting (N - a number):
    let L be a list of numbers;
    extend L to N entries;
    repeat with X running from 1 to N:
        now entry X of L is X;
    say "L unrandomised is [L].";
    sort L in random order;
    say "L randomised is [L].";
    sort L;
    say "L in ascending order is [L]."
```

builds a list of N numbers (initially all 0), fills it with the numbers 1, 2, 3, ..., N, then randomly reorders them, then sorts them back again, recovering the original order. The text produced by "check sorting 10" depends partly on chance but might for instance be:

```
L unrandomised is 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.
L randomised is 6, 2, 9, 3, 10, 1, 7, 4, 8 and 5.
L in ascending order is 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.
```

As with text in the previous chapter, a project which needs really long lists should use the Glulx virtual machine - "check sorting 10000", for instance, would break the default memory environment on the Z-machine, which is very tight, but works fine (if not very rapidly) on Glulx.

Back to §21.9. Accessing entries in a list
Onward to §21.11. Variations: arrays, logs, queues, stacks, sets, sieves and rings

Example 430: ★ **Leopard-skin** A maze that the player can escape if he performs an exact sequence of actions.

Example 431: ★★ **The Facts Were These** Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

---

§21.11. Variations: arrays, logs, queues, stacks, sets, sieves and rings

Lists are highly adaptable, and many other collection-like constructions can be made using them. This section introduces no new material, but simply suggests some of the variations which are possible.

1. The traditional computing term **array** means a list of values accessed by their entry numbers, often used in mathematical computations. The difference between an array and a list is mostly one of attitude, but usually arrays are fixed in length whereas lists can expand or contract.

2. A **log** is a list which records the most recently arrived values, but does not allow itself to grow indefinitely. In the following, which remembers the seven most recently taken items, new values arrive at the end while old ones eventually disappear from the front:

```
The most-recently-taken list is a list of objects that varies.
Carry out taking something (called the item):
    truncate the most-recently-taken list to the last 6 entries;
    add the item to the most-recently-taken list.
After taking:
    say "Taken. (So, your recent acquisitions: [most-recently-taken
list].)"
```

Note that the most-recently-taken list begins play as the empty list, grows as the first few items are taken, but then stabilises at length 7 thereafter. If we need to remember recent history, but only *recent* history, then a log is better than a list which can grow indefinitely, because there is no risk of speed reduction or memory exhaustion in a very long story.

3. A **queue** is a list of values which are waiting for attention. New values join at the back, while those being dealt with are removed from the front (whereupon the whole queue moves up one). An empty queue means that nobody is waiting for attention: but there is, in principle, no upper limit to the size of a queue, as anyone who has tried to make a couchette reservation at Roma Termini will know.

Queues typically form when two independent processes are at work, but going at different or variable speeds. An empty queue looks just like any other list:

```
The queue is a list of objects that varies.
```

(Invariably people, in what follows, but we'll make it a "list of objects" to allow for other possibilities too.) Once we identify a "new customer", we can join him to the queue thus:

```
add the new customer to the queue;
```

The process of serving the customers needs to make sure there is actually somebody waiting in the queue before it does anything:

```
Every turn when the number of entries in the queue is not 0:
    let the next customer be entry 1 of the queue;
    say "[The next customer] is served and leaves.";
    remove entry 1 from the queue.
```

Of course queues can also be constructed which empty from other positions, rather than the front: or we could make what computer scientists sometimes call a **deque**, a "double-ended queue" where new values arrive at both ends.

4. A **stack** is like a queue except that values arrive at, and are removed from, the same end. Stacks are slightly faster if the active end is the back rather than the front, though this will only be noticeable if they grow quite large.

To put a value V onto a stack S (which is known as "pushing") is simple:

```
add V to S;
```

And to remove a value from the top of the stack (which is known as "pulling"):

```
let N be the number of entries in S;
let V be entry N of S;
remove entry N from S;
```

Note that the middle line, accessing entry N, will fail if N = 0, that is, if the stack is empty: Inform's list routines will produce a run-time problem message.

Stacks are useful if some long-term process is constantly being interrupted by newer and more urgent demands, but they can also be used in planning. If a character has a long-term goal, which needs various short-term goals to be achieved along the way, then a stack can represent the goals currently being pursued. The top of the stack represents what the character is trying to achieve now. If the character realises that it needs to achieve something else first, we put that new goal onto the top of the stack, and it becomes the new current goal. When the character completes a task, it can be removed, and we can go back to trying to finish whatever is now on top. When the stack is empty, the character has achieved the original goal.

5. Notoriously, **set** has 464 distinct meanings in the Oxford English Dictionary, making it the single most ambiguous word in the language. Here we mean not the home of a badger or the Egyptian god of the desert, but the mathematical sense: a collection of values (sometimes called "elements") without duplicates, and which is normally written in brace notation and in some natural order for the reader's convenience.

The trick here is to maintain the principle that, at all times, our list is sorted in order and contains no duplicates. To provide an example, we start with two sets of numbers:

```
let S be {2, 4, 8, 16, 32, 64};
let T be {2, 4, 6, 10};
```

Here we add an element to T:

```
add 8 to T, if absent; sort T;
```

The "if absent" clause ensures that no duplicate can occur, and by sorting T afterwards, we maintain the principle that a set must remain in order - so T is now {2, 4, 6, 8, 10}, not {2, 4, 6, 10, 8}. (Inform's sorting algorithm is fast on nearly-sorted lists, so frequent sorting is not as inefficient as it might look.)

We next take the union of T and S, that is, the set containing everything which is in either or both:

```
let U be S; add T to U, if absent; sort U;
```

This makes U = {2, 4, 6, 8, 10, 16, 32, 64}, and once again no duplicates occur and we preserve the sorting. The intersection of T and S, the set of elements in both of them, is a little trickier:

```
let I be T;
repeat with the element running through T:
    if the element is not listed in S, remove the element from I.
```

(Faster methods could be devised which exploit the sortedness of T and S, but are not worth it for shortish lists.) This produces I = {2, 4, 8}. Lastly, we can form the set difference, consisting of those elements which are in S but not in T:

```
let D be S; remove T from D, if present;
```

Here, as with intersection, since all we do is to strike out unwanted elements, the surviving ones remain in order and there is no need to sort when we are finished. This produces D = {16, 32, 64}.

6. A **sieve** is used to make a complicated choice where there are many constraints, by ruling out impossible cases to see what is left. The term derives from the kitchen utensil (for sieving fine grains of flour), but via the name of the "sieve of Eratosthenes", an ancient Greek method for determining the prime numbers.

Using a sieve is much like using a set, and the difference is mainly one of outlook - we are interested in what does not belong, rather than what does.

7. A **ring** is not so much a row of values, more a circle, with the last and first entries thought of as adjacent. One position is usually thought of as special, and is the place where new items are added: this may as well be entry 1. For instance, to add "new

item" to the ring:

```
add the item at entry 1 in the ring;
```

To set "item" to the frontmost value and extract it from the ring:

```
let the item be entry 1 of the ring;
remove entry 1 from the ring;
```

And we can rotate the ring in either direction, making a different entry the new entry 1 and therefore the new frontmost value:

```
rotate the ring;
rotate the ring backwards;
```

A last note to conclude the chapter on lists. Lists, like almost all other values in Inform, can be passed to phrases as parameters. However, note that they are genuine values, not what some programming languages call "references" or "pointers". So the following:

```
To mess with (L - a list of numbers):
    add 7 to L, if absent.
```

does nothing, in practice. If given a list, it adds 7 to the list, but then throws it away again, so the longer list is never seen; it's exactly like

```
To mess with (N - a number):
    now N is 3.
```

which can never affect anything other than its own temporary value "N", which expires almost immediately in any case.

If we want a phrase which changes a list in a useful way and gives it back to us, we need a phrase which both takes in and gives back:

```
To decide which list of numbers is the extended (L - a list of
numbers):
```

```
add 7 to L, if absent;
decide on L.
```

And then, for example -

```
the extended { 2, 4, 6 };
```

produces:

```
{ 2, 4, 6, 7 }
```

(This may seem surprising since Inform's built-in phrases to adjust lists, like "add ... to ...", don't work this way. But those are written using Inform 6 code; see the phrase definitions in the Standard Rules for more.)

Start of Chapter 21: Lists

Back to §21.10. Lengthening or shortening a list

Onward to Chapter 22: Advanced Phrases: §22.1. A review of kinds

Example 432: **Circle of Misery** Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

Example 433: **Eyes, Fingers, Toes** A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

Example 434: **The Fibonacci Sequence** The modest Leonardo Fibonacci of Pisa will be only too happy to construct his

sequence on request, using an array.

Example 435: **I Didn't Come All The Way From Great Portland Street** In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

Example 436: **Lugubrious Pete's Delicatessen** In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

Example 437: **Sieve of Eratosthenes** The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

Example 438: **Your Mother Doesn't Work Here** Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

Examples from Chapter 21: Lists

Start of this chapter

Chapter 22: Advanced Phrases

425 Example Oyster Wide Shut

RB

Replacing Inform's default printing of properties such as "(closed)", "(open and providing light)", etc., with our own, more flexible variation.

As we've seen in earlier examples such as "Equipment List", it is possible to vary the way Inform creates inventory listings in general (to create lists that look more like paragraphs of prose, lists divided between what the player is wearing and what he isn't, and so on). We can also use activities to alter the printing of specific objects' names and contents, as with the "omit contents in listing" feature after printing the name of something.

We may find, however, that we would like a great deal more control over Inform's printing of inventory details, not just as a special effect for a few items, but throughout the game.

We start by turning off Inform's native property writer:

```
"Oyster Wide Shut"

Section 1 - Procedure

The print standard inventory rule is not listed in any
rulebook.

Carry out taking inventory (this is the new print inventory
rule):
    say "You are carrying: [line break]";
    list the contents of the player, with newlines,
indented, including contents, with extra indentation.
```

This is very much like the library's standard behavior, but with the exception that "giving inventory information" or even "giving brief inventory information" are omitted. Here's how we supplant it:

```
After printing the name of something (called target)
while taking inventory:
    follow the property-aggregation rules for the target.
```

Now, our property-aggregation rulebook is going to look at a given object and decide on a list of features that should be mentioned in inventory. We'll start by producing something quite similar to Inform's default behavior:

```
The property-aggregation rules are an object-based
rulebook.
The property-aggregation rulebook has a list of text
called the tagline.


A first property-aggregation rule for an openable open
thing (this is the mention open openables rule):
    add "open" to the tagline.


A first property-aggregation rule for an openable closed
thing (this is the mention closed openables rule):
    add "closed" to the tagline.


A property-aggregation rule for a closed transparent
container which contains nothing (this is the mention
empty transparent containers rule):
    add "empty" to the tagline.


A property-aggregation rule for an open container which
contains nothing (this is the mention empty open
containers rule):
    add "empty" to the tagline.


A property-aggregation rule for a lit thing (this is the
mention lit objects rule):
    add "providing light" to the tagline.
```

```
A property-aggregation rule for a thing worn by the
player (this is the mention worn objects rule):
    add "being worn" to the tagline.


The last property-aggregation rule (this is the print
aggregated properties rule):
    if the number of entries in the tagline is greater than
0:
        say " ([tagline])";
        rule succeeds;
    rule fails.
```

Notice that we don't need to write any rules about how to print that list of text: because Inform is printing out a list, it will automatically insert commas, spaces, and the word "and" where appropriate; and it will automatically follow the "use serial comma" option, if we have it set.

Now we're free to meddle. Let's give the player a bunch of possessions that will be listed in interesting ways in inventory:

```
Section 2 - Scenario


The Curved Beach is a room. "White sand stretches
away both northeast and northwest, enclosing this
attractive little bay. Gentle waves lap at the beach."


The player carries a glowing plastic sack. The glowing
plastic sack is lit and transparent and openable and
open. It contains a rock. It is wearable.


The player wears a flashlight lanyard. The flashlight
lanyard is a device.


Carry out switching on the lanyard: now the noun is lit.
```

```
Carry out switching off the lanyard: now the noun is
unlit.
```

```
The player carries an oyster. The oyster contains a
pearl. The oyster is openable.
```

Now suppose that we don't want the oyster to say "closed" when it's closed. Instead, we'd like it to say "clamped shut". As this is the only property the oyster will ever have, we can simply override his whole property-aggregation rulebook:

```
A property-aggregation rule for the oyster:
    if the oyster is closed:
        say " (clamped shut)";
    rule succeeds.
```

That's fine for the oyster because "clamped shut" is the only property he'll ever have. What if we'd like instead just to revise the way the sack (and only the sack) gets described as providing light?

```
The sacklight rule is listed after the mention lit objects
rule in the property-aggregation rules.
```

```
A property-aggregation rule for the plastic sack (this is
the sacklight rule):
    if "providing light" is listed in the tagline:
        remove "providing light" from the tagline;
        add "gently glowing" to the tagline.
```

Now the flashlight (and any other regular light sources we might add to the game) will be described as "providing light", but the sack will only be said to be gently glowing -- a bit more appropriate for its rather fainter gleam.

We might also wish to add a systematic feature across the board to include a new property in the inventory list? Let's say the player can enchant his possessions, and enchanted possessions should thereafter be listed accordingly:

```
A thing can be magical or non-magical. A thing is
usually non-magical.

Understand "enchant [something]" as enchanting.
Enchanting is an action applying to one thing.

Carry out enchanting something:
    now the noun is magical.

Report enchanting something:
    say "Ding! You turn [the noun] magical."

A property-aggregation rule for a magical thing:
    add "enchanted" to the tagline.

Test me with "i / close oyster / i / turn on flashlight / i /
take off flashlight / i / turn off flashlight / i / close sack / i /
open sack / i / take all from sack / i / close sack / i / wear
sack / i / enchant sack / i / open sack / put all in sack / i /
close sack / i".
```

Further variations are possible as well: if we used a "before printing the name..." rather than an "after printing the name..." rule, we could automatically generate lines like "an open and empty phosphorescent plastic sack", removing some of the artificiality of the parentheses.

Or we could add more logic to the rules about which properties are mentioned, so that some features of objects were mentioned in inventory only if the player was wearing the correct detection device, like so:

```
The player wears enchantment-detecting goggles.

A property-aggregation rule for a magical thing:
    if the player is wearing the goggles:
```

add "enchanted" to the tagline.

Example: Robo 1

RB

A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

"Robo"

The Experimentation Chamber is a room. Robo is a man in the Experimentation Chamber. "Robo, your prototype tin companion, stands awkwardly beside you. In the middle of his chest is a red enamel button[if the red button is switched on], currently depressed[otherwise], currently un-depressed[end if]."

The red button is a device. It is part of Robo. Instead of pushing the red button: if the red button is switched off, try switching on the red button; otherwise try switching off the red button.

After switching on the red button:
    say "CLICK! Robo is now in play-back mode."

After switching off the red button:
    say "CLACK! Robo is now in observation mode."

Definition: Robo is watching if the red button is switched off.

The current instruction set is a list of stored actions that varies.

After doing something when Robo is watching and Robo can see the player:
    now the actor is Robo;
    add the current action to the current instruction set;
    now the actor is the player;
    say "Robo watches you [the current action][one of], his yellow eyes lamp-like and observant[or]. In his metal head, gears whirr[or], his brushed-copper handlebar moustaches twitching[or] impassively[at random].";
    continue the action.

Every turn when Robo is not watching:
    if the number of entries in the current instruction set is 0:
        say "Robo has run out of behavior and grinds to an (expectant) halt.";
        now the red button is switched off;
    otherwise:
        let the next task be entry 1 of the current instruction set;
        try the next task;
        remove entry 1 from the current instruction set.

The red block and the blue cylinder are things in the Experimentation Chamber. The counter is a supporter in the Experimentation Chamber. The counter is scenery.

Report Robo examining Robo:
    say "Robo examines each of his hands in turn, then each of his legs (bending over mostly double in the middle to do this)." instead.

Report Robo examining the player:
    say "Robo stares at you, unblinkingly, for several seconds together[if a random chance of 1 in 7 succeeds]. His left moustache-bar twitches

```
infinitesimally upward[end if]." instead.

Report Robo taking the cylinder:
    say "[one of][Robo] needs several attempts to get his
metal fingers around [the cylinder] -- they are not
designed for grasping small objects elegantly. But at last
he succeeds[or]Once again, Robo struggles a bit before
picking up [the cylinder][stopping]." instead.

Test me with "z / take cylinder / take block / put cylinder
on counter / put block on counter / x robo / x me / get
block / drop block / press red button / z / z / z / z / z / z /
z / z / z / z".
```

---

427 **Example** **What Makes You Tick** *RB*

Building a fishing pole from several component parts that the player might put together in any order.

Suppose we want to let the player build a fishing pole out of three parts: a hook, a string, and a stick.

There are several things we must account for here. One is that our combination verb should be insensitive to ordering: it shouldn't matter whether the player types COMBINE STICK WITH STRING or COMBINE STRING WITH STICK.

Second, we need to make sure that our implementation handles intervening stages of assembly gracefully. The player should be able to combine string and hook first, or string and stick first, and be able to complete the assembly in either case.

Our implementation here uses a table of lists to determine which combinations of inputs should produce which result object. Because we sort our lists before comparing them, we guarantee that the player's ordering doesn't matter: COMBINE STICK WITH STRING will have the same effect as COMBINE STRING WITH STICK.

What's more, our implementation could be expanded to account for many other assemblages, if we wanted object-building to be a running theme of puzzles in our game.

"What Makes You Tick"

Understand "combine [something] with [something]" as combining it with. Combining it with is an action applying to two carried things. Understand the command "connect" as "combine".

Understand the command "attach" as something new. Understand "attach [something] to [something]" as combining it with.

The combining it with action has an object called the item built.

Setting action variables for combining something with something:
    let X be a list of objects;
    add the noun to X;
    add the second noun to X;
    sort X;
    repeat through the Table of Outcome Objects:
        let Y be the component list entry;
        sort Y;
        if X is Y:
            now the item built is the result entry.

Check combining it with:
    if the item built is nothing or the item built is not in limbo,
        say "You can't combine [the noun] and [the second noun] into anything useful." instead.

Carry out combining it with:
    move the item built to the holder of the noun;

```
    now the noun is nowhere;
    now the second noun is nowhere.


Report combining it with:
    say "You now have [an item built]."


Limbo is a container. Limbo contains a hookless fishing
pole, a hooked line, and a complete fishing pole.


Streamside is a room. The player carries a stick, a wire
hook, and a string.


Table of Outcome Objects

result          component list
hookless fishing pole   {stick, string}
hooked line     {wire hook, string}
hooked fishing pole     {hooked line, stick}
complete fishing pole   {hookless fishing pole, wire hook}


Test me with "combine stick with string / i / combine
pole with hook / i".
```

This kind of implementation makes sense if we don't intend the player to take the fishing pole apart again, or to refer to any of its component parts once it is built. For an alternate approach that does allow assembled objects to be taken apart again, see "Some Assembly Required".

ExampleFormicidae *RB*

Manipulating the order in which items are handled after TAKE ALL.

Suppose we have an item that produces an interesting result the first

time the player lifts it -- a rock with dangerous ants revealed underneath. The effect of the surprise is a little weakened, though, if the player sees that response as the result of a TAKE ALL, when it might be printed like this:

```
>[3] get all
tent peg: Taken.
water flask: Taken.
trading permit: Taken.
innocent-looking rock: You reach for the rock and turn it
over to reveal a thriving colony of flesh-eating ants.
Needless to say, you drop the rock and jump back with
a decidedly effeminate scream. They can probably hear
you all the way back in the base camp.
rusty nail: Taken.


[Your score has just gone down by two points.]
```

The calm response to "rusty nail" looks odd now, and the score change is disconnected from the event that caused it.

To manage this, we might institute a system so that interesting objects are handled last in their list, like so:

```
"Formicidae"


Use scoring.


Section 1 - Procedure


The magic rule is listed before the generate action rule
in the turn sequence rules.


A thing has a number called dramatic potential.
```

This is the magic rule:
    let L be the multiple object list;
    if the number of entries in L is greater than 1:
        sort L in dramatic potential order;
        alter the multiple object list to L.

Section 2 - Scenario

The Foothills is a room. "The land has become hilly; though the soil is still mostly coarse yellow sand, clumps of grass are able to grow in the shadier places. Deep wagon ruts running from the southwest towards the mountains in the northeast show where generations of caravans have already passed."

The water flask, the tent peg, and the trading permit are things in Foothills.

The rock is a thing in Foothills. Before printing the name of the rock when the rock is not handled: say "innocent-looking ". The dramatic potential of the rock is 10.

The rusty nail is a thing in Foothills.

The ant colony is a fixed in place thing. "A busy group of ants are crawling to and fro in the unaccustomed sun." Rule for deciding whether all includes the ant colony while taking: it does not.

Instead of taking the rock when the rock is handled:
    say "It might still have a stray ant or two on it."

```
After taking the rock:
    now the rock is handled;
    move ant colony to the location;
    move the rock to the location;
    say "You reach for the rock and turn it over to reveal
a thriving colony of flesh-eating ants. Needless to say,
you drop the rock and jump back with a decidedly
effeminate scream. They can probably hear you all the
way back in the base camp.";
    decrease score by 2.


Test me with "get peg / drop peg / get all / get rock".
```

Note that while one could also manipulate the object list to add or remove items at this stage, there's a simpler way to control what Inform considers "ALL" to mean in commands: see the activity "Deciding whether all includes" in the activities chapter.

---

429 ExampleRobo 2 *RB*

A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

We have seen how we can make a robot that watches the player, then plays back the same actions again. A slightly more adventurous implementation would be to let the player create a whole series of named scripts which the robot will run on command.

To do this, we'll need each program to have a command that sets it off (stored as text, since this is the best way to capture and preserve arbitrary text entered by the player) and then the script of actions that must result:

```
"Robo 2"


Use dynamic memory allocation of at least 16384.
```

```
Chapter 1 - Programming

Section 1 - The Programs Themselves

The hard drive is a container. A program is a kind of
thing. 15 programs are in hard drive. A program has
some text called the starter command. A program has a
list of stored actions called the script. Understand the
starter command property as describing a program.

Rule for printing the name of a program (called the
target) which is not blank:
    say "[starter command of the target in upper case]".

Definition: a program is blank if the number of entries in
its script is 0.

The current instruction name is some text that varies.
The current instruction set is a list of stored actions that
varies.
```

Now, we want to let Robo learn new programs; for this purpose, we'll emulate the code from our previous implementation, so that Robo watches what the player does and stores those actions in his script:

```
Section 2 - Learning New Programs

Understand "learn [text]" as learning. Learning is an
action applying to one topic.
```

```
Check learning:
    say "You have already learned all you need to know.
Robo, however, remains to be trained." instead.


Check Robo learning:
    if Robo is watching, say "Robo is already recording
'[the current instruction name]'." instead.


Carry out Robo learning:
    truncate the current instruction set to 0 entries;
    now the current instruction name is the topic
understood;
    now Robo is watching.


Report Robo learning:
    say "'Learning [the current instruction name in upper
case],' Robo replies."


After doing something when Robo is watching and Robo
can see the player:
    now the actor is Robo;
    add the current action to the current instruction set;
    now the actor is the player;
    say "Robo watches you [the current action][one of],
his yellow eyes lamp-like and observant[or]. In his metal
head, gears whirr[or], his brushed-copper handlebar
moustaches twitching[or] impassively[at random].";
    continue the action.
```

Of course, we also need to be able to switch learning mode off, and store any script learned this way. We'll also use the same STOP command to make Robo terminate a program he's in the middle of running.

```
Section 3 - Returning to Standby Mode


Understand "stop" as stopping. Stopping is an action
```

```
applying to nothing.


Check stopping:
    say "The command is useful only for Robo." instead.


Check Robo stopping:
    if Robo is standing by, stop the action.


Carry out Robo stopping when Robo is watching:
    let N be a random blank program;
    if N is a program:
        now the starter command of N is the current
instruction name;
        now the script of N is the current instruction set;
        say "'Stored [current instruction name in upper
case].'";
    otherwise:
        say "FAILURE: no program slots remaining."


Carry out Robo stopping:
    now Robo is standing by.


Report Robo stopping:
    say "Robo is now standing by."
```

Next, we need to be able to play these programs back again. We'll give Robo a "current program" to store which program he's currently working on, and a number called "stage" which will record where he is in the script. Our previous implementation simply had Robo erase entries from his script list as he performed them, but this time we would like Robo to be able to remember and rerun the same scripts over and over, so we need something a little more subtle.

```
Section 4 - Running Learned Programs
```

```
Understand "run [any program]" as running. Running is
an action applying to one visible thing.


Check running:
    say "Only Robo can perform Robo's programs."
instead.


Check Robo running:
    if Robo is not standing by, stop the action.


Unsuccessful attempt by Robo running:
    say "'ERROR: Robo can launch new programs only
when on standby.'"


Carry out Robo running:
    now the current program of Robo is the noun;
    now the stage of Robo is 1;
    now Robo is replaying.


Report Robo running:
    say "'Running [the starter command in upper case],'
Robo confirms."


Every turn when Robo is replaying:
    let the chosen script be the script of the current
program of Robo;
    let maximum be the number of entries in the chosen
script;
    let N be the stage of Robo;
    let the next step be entry N of the chosen script;
    try the next step;
    increment the stage of Robo;
    if the stage of Robo is greater than the maximum:
        say "Robo's program ends, and he reverts to
stand-by mode.";
        now Robo is standing by;
```

```
now the stage of Robo is 1.
```

For the player's sanity, we should also provide a way to find out which programs Robo has stored in memory and what they do, so we design two listing commands:

```
Section 5 - Listing Learned Programs

Understand "list programs" as requesting program list.
Requesting program list is an action applying to nothing.

Check requesting program list:
    say "You will have to ask Robo to list programs."
instead.

Carry out Robo requesting program list:
    say "'The available program[if more than one program
is not blank]s[end if] [is-are list of programs which are
not blank].'".

Understand "describe [any program]" or "list [any
program]" as requesting script of. Requesting script of is
an action applying to one visible thing.

Check requesting script of:
    say "You will have to ask Robo to give you the script."
instead.

Carry out Robo requesting script of:
    say "The script of [noun] is: [script of the noun]."
```

And to complete the suite, in case the player runs into Robo's fifteen-program limit:

```
Section 6 - Deleting Learned Programs

Understand "delete [any program]" as deleting. Deleting
is an action applying to one visible thing. Understand
the command "erase" as "delete".

Check deleting:
    say "You will have to instruct Robo to delete [the
noun]." instead.

Check Robo deleting (this is the can't delete except in
standby rule):
    if Robo is not standing by, stop the action.

Unsuccessful attempt by Robo deleting:
    say "'ERROR: programs may only be deleted while
Robo is in stand-by mode.'" instead.

Carry out Robo deleting:
    truncate the script of the noun to 0 entries;
    now the starter command of the noun is "".

Report Robo deleting:
    say "'Program deleted.'"
```

Now we use pretty much the same set-up as before to test Robo's abilities:

```
Chapter 2 - The Scenario
```

The Experimentation Chamber is a room.

Robo is a man in the Experimentation Chamber. "Robo, your prototype tin companion, stands awkwardly beside you[if watching], watching[end if]." Robo can be watching, replaying, or standing by. Robo is standing by. Robo has a program called the current program. Robo has a number called the stage.

Persuasion rule: persuasion succeeds.

The red block and the blue cylinder are things in the Experimentation Chamber. The counter is a supporter in the Experimentation Chamber. The counter is scenery.

Report Robo examining Robo:
    say "Robo examines each of his hands in turn, then each of his legs (bending over mostly double in the middle to do this)." instead.

Report Robo examining the player:
    say "Robo stares at you, unblinkingly, for several seconds together[if a random chance of 1 in 7 succeeds]. His left moustache-bar twitches infinitesimally upward[end if]." instead.

Report Robo taking the cylinder:
    say "[one of][Robo] needs several attempts to get his metal fingers around [the cylinder] -- they are not designed for grasping small objects elegantly. But at last he succeeds[or]Once again, Robo struggles a bit before picking up [the cylinder][stopping]." instead.

Test me with "test chocolate / test vanilla".

```
Test chocolate with "learn chocolate / stop / list
programs / Robo, learn chocolate / take red / put all on
counter / Robo, stop / Robo, list programs / Robo, run
chocolate / z / Robo, run chocolate / Robo, stop / z".
```

```
Test vanilla with "Robo, learn vanilla / take all / i / drop
all / x robo / x me / Robo, stop / Robo, list programs /
Robo, list vanilla / Robo, run vanilla / z / z / robo, delete
vanilla / robo, stop / robo, list vanilla / robo, delete
vanilla / robo, list programs".
```

We could also have written this so that Robo learns new scripts by accepting the player's instructions, so that

```
>ROBO, LEARN LIBRARY THEFT
>ROBO, TAKE BOOK
>ROBO, EAST
>ROBO, STOP
```

...would store the script 'library theft' with the actions taking the book and going east. The implementation there would have been mostly identical, except that instead of an "after doing something..." rule, we would have captured commands as we asked Robo to perform them, and added those to the command list in progress. The alternative code might look something like this:

```
Before Robo doing something other than stopping when
Robo is watching:
    add the current action to the current instruction set;
    say "'CHECK: [current action] added to script,' says
Robo." instead.
```

```
Unsuccessful attempt by Robo doing something when
Robo is watching:
    say "He does not actually perform the action."
```

Example **Leopard-skin** *RB*

A maze that the player can escape if he performs an exact sequence of actions.

Suppose (as in Infocom's Leather Goddesses of Phobos) that we have a maze that the player can escape only by performing the correct sequence of actions in the correct order. One way to do this would be to keep a list of the player's most recent actions, and see whether these match up with the combination we have established as the maze's solution.

For instance:

```
"Leopard-skin"

The Fur-Lined Maze is a room. "This seemingly endless
sequence of rooms is decorated in a tasteful selection of
exotic furs and gilded fixtures."

Clapping is an action applying to nothing. Understand
"clap" as clapping.
Kweepaing is an action applying to nothing. Understand
"kweepa" as kweepaing.

Carry out clapping:
    say "You clap."

Carry out kweepaing:
    say "You holler 'KWEEPA!' triumphantly."

The maze-sequence is a list of stored actions that
varies.
```

```
When play begins:
    add jumping to the maze-sequence;
    add clapping to the maze-sequence;
    add kweepaing to the maze-sequence.


The attempted-sequence is a list of stored actions that
varies.


Every turn when the player is in the Fur-Lined Maze:
    truncate the attempted-sequence to the last two
entries;
    add the current action to the attempted-sequence;
    if the attempted-sequence is the maze-sequence:
        say "That does it! You are instantly transported
from the maze!";
        end the story finally.


Test me with "hop / clap / clap / hop / kweepa / hop /
clap / kweepa".
```

---

431 **Example** **The Facts Were These** *RB*

Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

Occasionally it happens that we want to process an action on multiple items differently than we would if the player had just typed each of the individual actions separately. In this example, the reason is that we can only successfully GIVE items when their combined value passes a certain threshold amount; otherwise the recipient will reject them.

This works as an implementation of money, if we give value only to cash objects (though several other implementations of cash are available, most of which are simpler and more efficient). We could also imagine a mechanic like this being used for a bargaining or auction game as well, given a society that deals in objects rather than credits.

In order to consider all the items in the gift at once, we create an action that applies to multiple objects, but will in fact test the whole object collection during the first pass and print a definitive answer to whether the action succeeded. All subsequent times the game consults the rulebook will be stopped at the very beginning. No further processing will occur or output be printed.

```
"The Facts Were These"

Section 1 - Procedure
```

We start by creating the idea that everything in the game has a monetary value:

```
A price is a kind of value. $10 specifies a price. A thing
has a price.

Understand "give [things preferably held] to [someone]"
as multiply-giving it to. Understand "give [things] to
[someone]" as multiply-giving it to. Multiply-giving it to is
an action applying to two things.
```

A subtlety here: we say "things preferably held" to prefer items that the player is holding (so if the player has two dollars in hand and a third lies on the ground, he will use just the two he has).

The second grammar line allows Inform to match things that aren't held if it can't make up the list from things that are. If all three dollars are on the ground, the player can pick them up before spending them.

We do not, however, make multiply-giving apply to a "carried" item, because that will generate implicit takes of those items in a way that will mess up our action reporting. Instead, we're going to build the implicit takes into the system in a different way, one that permits us to collate the reports more attractively and print a short, one-sentence list of anything that the player had to pick up.

```
A thing can be given or ungiven. A thing is usually
```

```
ungiven.
```

This is for record-keeping purposes so that we can print an attractive list of what was given at the end of the turn.

```
First check multiply-giving it to:
    if already gave at the office is true:
        stop the action.
```

```
Already gave at the office is a truth state that varies.
```

"Already gave at the office" is the perhaps-excessively-named flag that keeps track of whether we've already done this action once.

```
Check multiply-giving something to the player:
        now already gave at the office is true;
        say "You can hardly bribe yourself.[paragraph
break]" instead;
```

The following rule is longish because it processes the entire list at once, generating implicit takes if necessary (but processing those implicit takes silently according to its own special rule, so that the output can be managed attractively). We are also, at the same time, calculating the total value of the player's offer.

```
Check multiply-giving it to:
    let L be the multiple object list;
    let bribe-price be $0;
    repeat with item running through L:
        if the player does not carry the item:
            abide by the ungivability rules for the item;
            carry out the implicitly taking activity with the
item;
            if the player does not carry the item:
                now already gave at the office is true;
                say "You can't include [the item] in your bribe,
since you're not holding [them]![paragraph break]"
instead;
        increase bribe-price by the price of item;
    if the number of entries in the recently-collected list is
greater than 0:
```

```
        repeat with item running through the recently-
    collected list:
            now item is marked for listing;
        say "You pick up [the list of marked for listing
    things] and make your offer. [run paragraph on]";
        now everything is unmarked for listing;
      if the bribe-price is less than the price of the second
    noun:
        now already gave at the office is true;
        say "[The second noun] angrily rejects your piffling
    bribe.[paragraph break]" instead.
```

The bit about making some items "marked for listing", above, rather than printing the list directly, is that using the "[the list of....]" syntax guarantees that Inform will respect grouping rules in writing its description. For instance, if the player has automatically taken all three dollars, the output will say "the three dollars" instead of "the dollar, the dollar, and the dollar."

```
Carry out multiply-giving it to:
    let L be the multiple object list;
    repeat with item running through L:
        now the second noun carries the item;
        now the item is given;
    now already gave at the office is true;
```

```
Report multiply-giving it to:
    say "[The second noun] rather shamefacedly tucks
[the list of given things] away into a pocket.[paragraph
break]".
```

Now we create our own variation of implicitly taking in order to customize the output for the multiply-giving action. The "ungivability rules" should disallow any object that the player absolutely cannot take, because we want "carry out the implicitly taking activity" to succeed every time -- and therefore not print out any less-attractive results from implicit takes that don't succeed. Otherwise, the player's GIVE TREE AND DOG TO ATTENDANT might produce the reply "That's fixed in place" -- without specifying which object is fixed in place.

Because of the way this works, we will want to be careful: if we have any "instead of taking..." rules for special objects in the game, we should be sure to mirror those with an ungivability rule to print something more suitable in the case that the player tries taking that object as part of the multiple giving action.

The ungivability rules are an object-based rulebook.

An ungivability rule for a person:
    now already gave at the office is true;
    say "Slavery is illegal.[paragraph break]" instead.

An ungivability rule for something (called the item) which is enclosed by someone who is not the player:
    now already gave at the office is true;
    say "[The item] [aren't] yours to give.[paragraph break]" instead.

An ungivability rule for something which encloses the player:
    now already gave at the office is true;
    say "You don't want to end up as part of the gift.[paragraph break]" instead;

An ungivability rule for something (called the item) which is part of something:
    now already gave at the office is true;
    say "[The item] [are] attached to [a random thing which incorporates the item][paragraph break]" instead.

An ungivability rule for something (called the item) which is scenery:
    now already gave at the office is true;
    say "[The item] [are] unremovable.[paragraph break]" instead.

An ungivability rule for something (called the item) which is fixed in place:
    now already gave at the office is true;
    say "[The item] [are] fixed in place.[paragraph break]"

```
instead.


An ungivability rule for a direction (called the item):
    now already gave at the office is true;
    say "[The item] [are] not susceptible to giving.
[paragraph break]" instead.


Rule for implicitly taking something (called target) while
multiply-giving:
    silently try taking the target;
    if the player carries the target:
        add the target to the recently-collected list.


The recently-collected list is a list of objects that varies.
```

And since we don't want to list the individual objects separately:

```
The selectively announce items from multiple object lists
rule is listed instead of the announce items from multiple
object lists rule in the action-processing rules.


This is the selectively announce items from multiple
object lists rule:
    if multiply-giving:
        do nothing;
    otherwise:
        if the current item from the multiple object list is not
nothing:
            say "[current item from the multiple object list]:
[run paragraph on]".
```

And now, since this ought to work symmetrically if the player provides just one high-value item:

```
Check giving something to someone:
    if the price of the noun is less than the price of the
second noun:
```

```
        say "[The second noun] angrily rejects your piffling
    bribe." instead.
```

As we've seen elsewhere, the giving action by default returns a refusal, but is also written to start working if we remove the blockage. So we do that here, and revise the report rule to match the report rule we have for multiple giving.

```
The block giving rule is not listed in any rulebook.

The new report giving rule is listed instead of the
standard report giving rule in the report giving it to rules.

This is the new report giving rule:
    say "[The second noun] rather shamefacedly tucks
[the noun] away into a pocket."
```

After each instance of the multiply-giving action, we need to clear the variables we used to track its state. We could do this in "Before reading a command", but that's unsafe because the player might type GIVE PIE AND CAP TO ATTENDANT. GIVE DOLLARS TO ATTENDANT. all on a single line, and we would like to be able to clear the variables between one action and the next. The correct place to attach this behavior is immediately before the generate action rule, thus:

```
The before-generation rule is listed before the generate
action rule in the turn sequence rules.

This is the before-generation rule:
    now every thing is ungiven;
    now already gave at the office is false;
    truncate the recently-collected list to 0 entries.

Section 2 - Scenario
```

The Morgue Office is a room. "This is not the Morgue itself; this is only its outer office. The familiar room full of silver drawers and cold air lies beyond."

The Morgue Attendant is a man in the Morgue Office. "The Attendant has seen you come through a number of times, and is becoming suspicious of your abiding interest in dead people." The description is "The Morgue Attendant is fifty-four years, six months, five days, and three minutes old." The price of the Morgue Attendant is $3.

A dollar is a kind of thing. The player carries three dollars. The price of a dollar is always $1.

The player carries a miniature rhubarb pie. The price of the miniature rhubarb pie is $5.

The player carries a knitted cap. The price of the knitted cap is $2.

Test me with "test dollars / purloin three dollars / test multi-line / purloin three dollars / purloin pie / purloin cap / test specificity / purloin three dollars / test largesse / test mixed-gift".

Test multi-line with "give dollar and pie to attendant. give dollars and cap to attendant".

Test dollars with "drop all / give dollar to Morgue Attendant / give dollars to Morgue Attendant / get dollars / give dollars to morgue attendant / purloin three dollars / drop dollars / give dollars to Morgue Attendant".

Test specificity with "give three dollars to Morgue Attendant".

Test largesse with "give pie to Morgue Attendant".

Test mixed-gift with "give dollar and cap to Morgue Attendant / get cap / give dollar and cap to morgue attendant / give me and dollar to attendant".

PURLOIN, used in the tests here, is a special debugging command that allows the player to acquire objects that wouldn't otherwise be possible to take. It is only active in non-release versions of the story. For more about debugging commands, see the chapter on Testing and Debugging.

---

432 Example Circle of Misery *RB*

Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

The only point we need to be careful about is that the carousel is simulated twice over, in the following text: once in the built-in way that objects are inside other objects, so that the luggage items are objects contained in the carousel object; but then again by the "circle of misery" list, a ring buffer keeping track of what order things are in. We need to be careful that these two records do not fall out of synchrony: anything put into the carousel must be added to the list, anything taken out must be removed. (In fact we forbid things to be added, for simplicity's sake.)

"Circle Of Misery"

Luggage item is a kind of thing. The blue golf bag, the leopardskin suitcase, the green rucksack and the Lufthansa shoulder-bag are luggage items.

```
Heathrow Baggage Claim is a room. The carousel is a
container in Heathrow. "The luggage carousel, a scaly
rubbered ring, does for the roundabout what Heathrow
Airport does for the dream of flight: that is, turns the
purest magic into the essence of boredom, only with
extra stress. [if the number of entries in the circle of
misery is 0]For once it stands idle. Perhaps it's broken.
[otherwise]The baggage approaching you now: [the
circle of misery with indefinite articles]."


The circle of misery is a list of objects that varies.


When play begins:
    now all the luggage items are in the carousel;
    add the list of luggage items to the circle of misery.
```

The list "circle of misery" is our ring, in which entry 1 is considered to be the position of whichever bag is currently frontmost. And here it goes, round and round:

```
Every turn when the number of entries in the circle of
misery is not 0:
    rotate the circle of misery;
    let the bag be entry 1 of the circle of misery;
    say "The carousel trundles on, bringing [a bag] to
within reach."


After taking a luggage item (called the bag):
    remove the bag from the circle of misery, if present;
    say "Taken."


Before doing something with a luggage item (called the
bag) which is in the carousel:
    if the bag is not entry 1 of the circle of misery, say
"[The bag] is maddeningly out of range. You'll have to
wait for it to come round." instead.
```

Instead of inserting something into the carousel, say "In recent years, the authorities have tended to frown upon depositing bags in random places at airports."

Test me with "get suitcase / get suitcase / get suitcase / get suitcase / look / get golf bag / look / get golf bag".

---

433 **Example** **Eyes, Fingers, Toes** *RB*

A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

It is not difficult to implement a safe which can be set to a single number to open; but a more common scenario in the real world is for the safe to open on a sequence of numbers when they have been dialed in the right order.

For IF, this means that we have to keep running track of the last N digits the player has dialed, dropping the first digit and adding a new one to the end each time the player re-dials the safe. This is a perfect occasion for lists:

"Eyes, Fingers, Toes"

The Addams Wine Cellar is a room. It contains a closed lockable locked container called a safe.

The safe has a list of numbers called the current combination.

The safe has a list of numbers called the true

combination. The true combination of the safe is {2, 10, 11}.

Understand "set [something] to [a number]" as setting it numerically to. Setting it numerically to is an action applying to one thing and one number.

Instead of examining the safe:
    if the number of entries in the current combination of the safe is 0,
        say "You haven't dialed the safe to any combination yet.";
    otherwise say "You have dialed the safe to [the current combination of the safe].".

Check setting something numerically to (this is the block setting numerically rule):
    say "[The noun] cannot be set."

Instead of setting the safe numerically to the number understood:
    truncate the current combination of the safe to the last 2 entries;
    add the number understood to the current combination of the safe;
    if the safe is locked and the current combination of the safe is the true combination of the safe:
        say "You dial [the number understood], and [the safe] gives a joyous CLICK.";
        now the safe is unlocked;
    otherwise if safe is unlocked and the safe is closed and the current combination of the safe is not the true combination of the safe:
        say "You spin the dial, and [the safe] snicks locked.";
        now the safe is locked;
    otherwise:
        say "You dial [the number understood] on the safe."

Test me with "x safe / set safe to 10 / x safe / set safe to 29 / x safe / set safe to 2 / x safe / set safe to 10 / x safe / set safe to 11 / open safe / set safe to 14 / close safe / set safe to 15 / open safe".

Example **The Fibonacci Sequence** *RB*

The modest Leonardo Fibonacci of Pisa will be only too happy to construct his sequence on request, using an array.

Fibonacci (a posthumous nickname) spread Arabic mathematical learning across Europe in the 13th century, and it's curious that his name lives on only for a single sequence.

"The Fibonacci Sequence"

Pisa is a room. Leonardo Fibonacci is a man in Pisa.
"The modest Italian mathematician, Leonardo Fibonacci (1170-1250), beams at you."

Sequencing is an action applying to one number.
Understand "sequence [number]" as sequencing.

Instead of sequencing, say "You make a feeble attempt, sketching in the sand, but it goes nowhere. Leonardo is sympathetic. 'Often goes wrong for me, too, actually. I didn't even invent the thing - the ancient Indians knew about it first.'"

Persuasion rule for asking Leonardo to try sequencing: persuasion succeeds.

```
Report Leonardo sequencing:
    let N be the number understood;
    say "Leonardo scratches his head and makes self-
deprecating remarks, before coming up with [the first N
terms of the Fibonacci sequence]."
```

An array need not be fixed in length, as the following example shows:

```
To decide what list of numbers is the first (F - a number)
terms of the Fibonacci sequence:
    let the Fibonacci sequence be {1, 1};
    let N be 3;
    while N < F:
        let the last term be entry (N - 1) of the Fibonacci
sequence;
        let the penultimate term be entry (N - 2) of the
Fibonacci sequence;
        let the next term be the last term plus the
penultimate term;
        add the next term to the Fibonacci sequence;
        increment N;
    decide on the Fibonacci sequence.



Test me with "sequence 20 / leonardo, sequence 20".
```

The result of "the first 20 terms of the Fibonacci sequence" is "1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584 and 4181". This is a sequence which has a knack of turning up in odd places - it was found in the 1970s to be related to the rings of florets in a sunflower, for instance - and here it is in a book about interactive fiction.

Example I Didn't Come All The Way From Great Portland Street *RB*

In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

There is very little to this, in fact. The tricky rule to enforce is Repetition: the player is forbidden to repeat any previously tried action. We keep track of this by keeping a set of past actions, which for want of a better term is called the "tally". All we need to do is:

```
        if the current action is listed in the tally, challenge for
        "Repetition of [the current action]!";
        otherwise add the current action to the tally.
```

Note that the tally can never contain duplicates, and that when, at the end of the round, we print it out, we sort it first - this makes a more natural-looking sentence. (Sorting a list of actions uses the natural order for actions: compare the sequence on the Actions page of the Index.) The full text, then, is:

```
        "I Didn't Come All The Way From Great Portland Street"

        The Paris Theatre is a room. An instrument is a kind of
        thing. The violin, the tuba, the xylophone and the
        triangle are instruments. The violin is inside the case.
        The tuba, the xylophone, the radish, the case, the bust
        of Nicholas Parsons, the purple felt hat and the triangle
        are in the Paris Theatre.

        The Round is a scene. The Round begins when play
        begins. The Round ends when the turn count is 10.

        The tally is a list of stored actions that varies.

        When the Round begins:
            say "'And the subject on the card is... musical
        instruments. Will you carry out for us something to do
        with that, please, for ten turns starting - now!'"

        When the Round ends:
            sort the tally;
            say "Phweeep![paragraph break]'So, when the
        whistle goes ten turns are up, you get a point for acting
        when the whistle blows, and in that round you
```

```
entertained us by [the tally], and you also get a bonus
point for keeping going until the whistle went.'";
    end the story finally.


To challenge for (infraction - text):
    say "Bzzzzt! 'And [one of]Clement Freud[or]Derek
Nimmo[or]Kenneth Williams[or]Peter Jones[at random]
has challenged.'[paragraph break]'[infraction]'[paragraph
break]'Well, as it's your first time playing the game, and
the audience was enjoying your contribution so much, I
will disallow the challenge, you have [10 minus the turn
count] turn[s] left on musical instruments, starting...
now!"


Before doing something:
    if the current action is listed in the tally, challenge for
"Repetition of [the current action]!" instead;
    otherwise add the current action to the tally;
    if waiting, challenge for "Hesitation!" instead;
    if not looking and not waiting and the noun is not an
instrument and the second noun is not an instrument,
challenge for "Deviation!" instead.


Test me with "look / wait / examine bust / take tuba / get
triangle / hit xylophone / get tuba / examine tuba / get
violin".
```

(The Paris Theatre in Lower Regent Street, London, was for many years the home of BBC radio panel games.)

Example **Lugubrious Pete's Delicatessen** *RB*

In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

First, to set the scene:

```
"Lugubrious Pete's Delicatessen"

The Supermarket is west of the Delicatessen Counter.
Lugubrious Pete is in the Delicatessen. "Lugubrious
Pete, dolefully slicing meats and cheeses, serves at the
counter." Alice, Beth, Gemma, Delia, and Eliza are
women in the Supermarket.

The deli queue is a list of objects that varies.
```

Two processes compete here: one that fills the queue, the other which will empty it. The first process is the one which brings shoppers in to the counter, joining the back of the queue, which is where "add ... to ..." puts new entries by default:

```
Every turn when a woman is in the Supermarket and a
random chance of 2 in 3 succeeds (this is the customer
arriving rule):
    let the customer be a random woman in the
Supermarket;
    now the customer is in the Delicatessen;
    if the player is in the Supermarket, say "[The
customer] walks into the Delicatessen.";
    if the player is in the Delicatessen, say "[The
customer] comes in from the Supermarket, and [if the
number of entries in the deli queue is 0]can't believe her
luck. The counter is free![otherwise]resignedly queues
behind [the deli queue].";
    add the customer to the deli queue.
```

The competing process is the one which serves shoppers and thus gets rid of them again: unfortunately, it is slower. But Pete is fair if inefficient, and serves the customers in strict order of arrival. Each served customer is removed from the front of the list, and the others therefore all move up a place.

```
Every turn when the number of entries in the deli queue
is not 0 and a random chance of 1 in 3 succeeds (this is
the customer being served rule):
    let the customer be entry 1 of the deli queue;
    if the player is in the Delicatessen, say "Pete gives a
```

```
droopy expression as he serves [customer], who
nevertheless brightens and leaves.";
    if the player is in the Supermarket, say "[customer]
emerges cheerfully from the Delicatessen Counter, and
goes about her regular shopping.";
    now the customer is in the Supermarket;
    remove entry 1 from the deli queue.


Instead of waiting in the Delicatessen when the number
of entries in the deli queue is not 0, say "Time passes,
for [deli queue] quite as much as for yourself."


Test me with "wait / wait / wait / east / wait / wait / wait /
wait / wait".
```

That completes the example, but here is a variation to show that queues need not empty from the front. The Deli already looks a pretty sexist establishment, with the customers all being women, but it is about to get a whole lot worse:

```
Modesty is a kind of value. The modesties are positively
prim, buttoned up, modest, flirty, revealing and
downright immodest. Every woman has a modesty.
Alice is positively prim. Beth is downright immodest.
Gemma is modest. Delia is flirty. Eliza is revealing.
```

We could then rewrite the service rule like so:

```
Every turn when the number of entries in the deli queue
is not 0 and a random chance of 1 in 3 succeeds (this is
the customer being served rule):
    let Pete's preference be the deli queue;
    sort Pete's preference in reverse modesty order;
    let the customer be entry 1 of Pete's preference;
    let the first in line be entry 1 of the deli queue;
    if the player is in the Delicatessen, say "[if the
customer is the first in line]Pete gives a droopy
expression as he serves [the customer], who
nevertheless brightens and leaves.
[otherwise]Outrageously, Pete scans the queue, notices
[the customer] in her [modesty of the customer] clothes,
```

```
and serves her next, while [the first in line] glares at
him.";
    if the player is in the Supermarket, say "[The
customer] emerges cheerfully from the Delicatessen
Counter, and goes about her regular shopping.";
    now the customer is in the Supermarket;
    remove the customer from the deli queue.
```

It is now heartbreakingly difficult for Alice to obtain her sliced chorizo sausage.

---

437 **Example** **Sieve of Eratosthenes** *RB*

The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

In the words of Wikipedia: "Eratosthenes of Cyrene (Greek Ερατοσθένης; 276 BC-194 BC) was a Greek mathematician, poet, athlete, geographer and astronomer." In the words of Tom Lehrer: "It's people like that who make you realise how little you've achieved in life."

A prime number is a number greater than 1 which is not a multiple of anything, so we can find the primes by starting with all the numbers and sieving out all the multiples of 2, then all the multiples of 3, and so on. Here we make our sieve of the unacceptable numbers (the "composite" or non-prime ones) first, then form a list of all the numbers, then sieve out the composites: what are left must be the primes.

```
"Sieve of Eratosthenes"


Alexandria is a room. Eratosthenes is a man in
Alexandria. "The haughty Greek mathematician,
Eratosthenes, glowers at you."


Sieving is an action applying to one number.
Understand "sieve [number]" as sieving.
```

```
Instead of sieving, say "You make a feeble attempt,
sketching in the sand, but it goes nowhere.
Eratosthenes smirks. 'I expect your friends call you
gamma, then?'"



Persuasion rule for asking Eratosthenes to try sieving:
persuasion succeeds.



Report Eratosthenes sieving:
    let N be the number understood;
    let the composites be a list of numbers;
    let I be 2;
    while I times I is at most N:
        if I is not listed in the composites:
            let J be I times 2;
            while J is at most N:
                add J to the composites, if absent;
                increase J by I;
        increment I;
    sort the composites;
    let the primes be a list of numbers;
    repeat with P running from 2 to N:
        add P to the primes;
    remove the composites from the primes;
    say "Eratosthenes sketches lines in the sand with the
air of much practice. 'The primes up to [N] are [the
primes]. The composites are [the composites].'"



Test me with "sieve 10 / eratosthenes, sieve 100".
```

While this could all be done more efficiently with an array, that's only because what we are sieving are numbers: sieving is a technique which can be used for non-numerical decisions, too.

Example **Your Mother Doesn't Work Here** *RB*

Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

Suppose the player's mother is supposed to be cleaning the living room,

but she can be interrupted by the need to pick up things the player has dropped. New tasks are added to the end of her "current plan" list; every turn, she attempts to do whatever is the last entry on that list.

"Your Mother Doesn't Work Here"

A person has a list of stored actions called the current plan.

Every turn:
    repeat with culprit running through people who are not the player:
        if the number of entries in current plan of the culprit is greater than 0:
            let N be the number of entries in the current plan of the culprit;
            try entry N of the current plan of the culprit;
            remove entry N from the current plan of the culprit.

The Living Room is a room. It contains a somewhat muddy Persian rug. Your mother is a woman in the Living Room.

West of the Living Room is the Kitchen.

Instead of your mother rubbing the rug:
    say "Your mother scrubs the stained areas of the rug, muttering to herself."

Instead of taking something:
    say "Nah, Mom'll get that."

```
Report your mother taking something:
    say "Your mother picks up [the noun][one of], sighing
deeply[or], jaw tight[or], with assorted comments on
your manners[or]; to judge from her comments, she is
also indulging in a pleasant fantasy about Swiss
boarding schools[stopping]." instead.


When play begins:
    add mother going west to the current plan of mother;
    add mother rubbing the rug to the current plan of
mother.


Every turn:
    if mother is not in the Living Room, end the story
finally.


Carry out dropping something:
    add mother taking the noun to the current plan of
mother.


The player carries some dirty socks, some dirty shoes, a
dirty hat, a pair of dirty trousers, and a backpack.


Test me with "drop socks / z / drop shoes / drop hat /
drop all / z / z".
```

As goal-seeking goes, this is fairly rudimentary; "Boston Cream" provides an alternative (and slightly more sophisticated approach), but for really complex goal-seeking characters, it is probably best to turn to the character extensions designed for Inform.

Chapter 22: Advanced Phrases

4. Default values for phrase kinds; *§22.5. Map, filter and reduce*; *§22.6. Generic phrases*; *§22.7. Kind va*

Contents of *Writing with Inform*

Chapter 21: Lists

Chapter 23: Figures, Sounds and Files

Indexes of the examples

§22.1. A review of kinds

Most of the time, what's created in an Inform source text will have a name which can be used as a value - sometimes openly so, sometimes not. In this book, we haven't gone out of our way to make that point, because there was no real need to do so. It's possible to make heavy use of rulebooks and write large-scale Inform projects without ever needing to use a rulebook's name as a value in its own right, for example. But if we want to create sophisticated extensions to Inform, or to use modern techniques such as functional and generic programming, we need to be fluent in the language of kinds.

Inform's language of kinds has four ingredients: base kinds, constructions, kind variables and kinds of kinds.

1. Base kinds. Inform provides the following base kinds for values:

```
object, number, real number, time, truth state, text, snippet, Unicode
character, action, scene, table name, equation name, use option,
action name, figure name, sound name, external file, rulebook
outcome, parser error
```

together with a few others, such as "response" and "verb", to do with linguistic features.

And Inform allows us to create new base kinds either by making more specialised kinds of object:

```
A geographical layout is a kind of object.
```

```
A marmoset is a kind of animal.
```

Or by making new enumerations or arithmetical kinds:

```
Distance is a kind of value. 10km specifies a distance.
Colour is a kind of value. Red, green and blue are colours.
```

2. Constructions. These are ways to make new kinds from existing ones. The construction most often used is "list of...". For any kind K, there is a kind called list of K. So the range of possible kinds in Inform is actually infinite, because:

```
number
list of numbers
list of lists of numbers
list of lists of lists of numbers
...
```

are all different from each other. Inform has nine constructions, as follows:

```
list of K
description of K
relation of K to L
K based rule producing L
K based rulebook producing L
activity on K
phrase K -> L
K valued property
K valued table column
```

Some of these have appeared in previous chapters, but in abbreviated form. For example, "rulebook" abbreviates "action based rulebook producing nothing", and "either/or property" is a synonym for "truth state valued property". The kinds of descriptions and phrases haven't been covered yet, but are coming up in the sections following.

These constructions can of course be combined:

```
phrase (relation of numbers to colours, number) -> list of colours
```

Brackets can be used to clarify matters:

```
phrase (phrase number -> (phrase number -> number)) -> nothing
```

Nothing will make that a simple idea, but it's unambiguous and can be puzzled out with practice.

3. Variables. In a way, that's everything: there are just base kinds and constructions on them, and those construct every possible kind in Inform. But the language we use to describe kinds is larger than that, because it allows us to describe many kinds at once, in the same way that Inform reads the word "something" as applying to many objects at once, not as a single object.

Kind variables will be covered later in the chapter, but the idea is that:

```
To hunt for (needle - value of kind K) in (haystack - list of Ks): ...
```

allows us to describe the kinds acceptable in a phrase so that a wide range of possibilities are allowed. The above matches both:

```
hunt for 4 in { 2, 3, 4, 5 };
hunt for "fish" in { "lemon sauce", "fish", "garden peas" };
```

The letter K in the definition stood for any kind; in the first use of "hunt" here, K turned out to be "number", and in the second it was "text". On the other hand Inform would reject:

```
hunt for 4 in { containment relation, support relation };
```

because there is no kind K which can make this match the definition.

There are potentially 26 kind variables, A to Z, though it's customary to use them in the order K, L, M, ..., and it's very rare to need more than two at a time.

4. Kinds of kind. Inform understands several names which look as if they are kinds, but actually aren't:

```
value, arithmetic value, enumerated value, sayable value
```

(Again, these are built in to Inform.) They are not kinds because they're just too loose and vague. Instead, they can be used in phrase definitions to match against multiple

possibilities:

```
To announce (X - sayable value): say "I declare that [X] has arrived."
```

This makes "announce X" work for any value X which can be said. All the same, "sayable value" is not a kind. It could never be safe for this to be the kind of a variable, because Inform would never know what could be done with the contents (except that it could be printed out).

5. Secret inner workings. There isn't a fifth ingredient, but if there were, it would be a handful of names used in matching some of the core built-in phrases of Inform which have so-called inline I6 definitions. These are not intended for anyone else to use, and are just an internal convenience; they aren't publicly documented and might change without notice. They don't describe kinds at all, because they tell the matcher to look for something else instead. For instance, there's one called "condition", which means "match a condition rather than a value". They appear in red ink in the Phrasebook index.

Start of Chapter 22: Advanced Phrases

Back to Chapter 21: Lists: §21.11. Variations: arrays, logs, queues, stacks, sets, sieves and rings

Onward to §22.2. Descriptions as values

§22.2. Descriptions as values

In the chapter on Descriptions, we saw that a description can be any source text which describes one or more objects: it might be as simple as "the Corn Market", or as complicated as "open containers which are in dark rooms". More or less the only restriction is that it must be unambiguous as to what counts and what does not: "three containers" is ambiguous as a description because it does not say which three.

We've now seen several interesting tricks with descriptions. In fact, if D is a description, then

```
say "You gaze mournfully at [the list of D].";
let the tally be the number of D;
let the surprise prize be a random D;
repeat with item running through D:
```

```
...
```

are all standard things to do. These examples make it look as if it must be possible to define phrases which act on descriptions, and in fact it is, because a description can be a value in itself. For example,

```
even numbers
open containers which are in dark rooms
```

are values of kind "description of numbers" and "description of objects" respectively. In general, if K is any kind then "description of K" is also a kind. Here is how we might make use of that:

```
To enumerate (collection - a description of objects):
    repeat with the item running through the collection:
        say "-- [The item]."
```

This makes "enumerate lighted rooms" run off a list of lighted rooms in a textual format different from the standard one produced by "say the list of lighted rooms". Inside the definition, "collection" can be used wherever a description might be used: here, for instance, we use it as the range for the repeat loop. (That's only possible because the range is limited in size: Inform wouldn't have allowed us to range through, say, all texts.)

Purely as a convenience, we can also write "member of" or "members of" in this context. For instance, in the enumerate definition, it would have been just as good to write "...running through the members of the collection..." in the repeat. (Similarly, we could write "number of members of the collection" or "a random member of the collection", which looks grammatically tidier than writing "number of the collection" or "random of the collection" - though in fact both of these do work.)

Finally, it's sometimes useful in an abstract situation to test

if (value) **matches** (description of values)**:**

This condition is true if the value matches the description; the kinds must be compatible, or Inform will issue a problem message. There is no point using this for cases where the description is given explicitly:

```
if 4 matches even numbers, ...
```

because it is easier to write just:

```
if 4 is an even number, ...
```

So this condition is only useful when the description is stored in some variable, and its identity is not known.

Start of Chapter 22: Advanced Phrases
Back to §22.1. A review of kinds
Onward to §22.3. Phrases as values
Example 439: **Curare** A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

§22.3. Phrases as values

Given any two kinds K and L, the kind "phrase K -> L" is now a kind. (This is meant to look like a mathematical function arrow.) For example, the phrase defined by

```
To decide which number is the square of (N - a number): ...
```

has the kind "phrase number -> number". Brackets and commas are used if the phrase combines several values, so

```
To decide which text is (T - text) repeated (N - a number) times: ...
```

has the kind "phrase (text, number) -> text". The word "nothing" is used if there are no values in, or no value out - thus

```
To decide which number is the magic target: ...
```

has kind "phrase nothing -> number", and

```
To dig (eastward - length) by (northward - length): ...
```

has the kind "phrase (length, length) -> nothing".

But how are we to get at these values? The answer is that we need to give a phrase a name in order to do so. For example:

```
To decide what number is double (N - a number) (this is doubling):
    decide on N plus N.
```

This is the same syntax used to name rules, and the idea is the same. If we

try "showme doubling", the result is

```
phrase number -> number: doubling
```

The main thing we want to do with a phrase is to apply it. So:

```
showme doubling applied to 2;
```

produces

```
"doubling applied to 2" = number: 4
```

There are versions of "applied to" for phrases applied to 0 to 3 values:

(phrase nothing -> value) **applied ... value**

This phrase produces the result of applying the given phrase, which must be one which takes no values itself.

(phrase value -> value) **applied to** (value) **... value**

This phrase produces the result of applying the given phrase, which must be one which takes one value itself.

(phrase (value, value) -> value) **applied to** (value) **and** (value) **... value**

This phrase produces the result of applying the given phrase, which must be one which takes two values itself.

(phrase (value, value, value) -> value) **applied to** (value) **and** (value) **and** (value) **... value**

This phrase produces the result of applying the given phrase, which must be one which takes three values itself.

So for example:

```
F applied
F applied to V
F applied to V and W
F applied to V and W and X
```

For phrases which do not produce a value, we use "apply":

apply (phrase nothing -> nothing)

This phrase causes the given phrase to be applied. It must be one which takes no values itself.

apply (phrase value -> nothing) **to** (value)

This phrase causes the given phrase to be applied. It must be one which takes one value itself.

apply (phrase (value, value) -> nothing) **to** (value) **and** (value)

This phrase causes the given phrase to be applied. It must be one which takes two values itself.

apply (phrase (value, value, value) -> nothing) **to** (value) **and** (value) **and** (value)

This phrase causes the given phrase to be applied. It must be one which takes three values itself.

Thus:

```
apply F;
apply F to V;
apply F to V and W;
apply F to V and W and X;
```

Start of Chapter 22: Advanced Phrases

Back to §22.2. Descriptions as values

Onward to §22.4. Default values for phrase kinds

§22.4. Default values for phrase kinds

The default value for "phrase K -> nothing" is a phrase which does nothing. For example, if we write:

```
let S be a phrase numbers -> nothing;
```

then S is created holding the default phrase numbers -> nothing, and if we then try it out with:

```
apply S to 17;
```

we will indeed find that nothing happens.

The default value for "phrase K -> L" is a phrase which, no matter what value of K it applies to, always produces the default value of L. (It's a sort of equivalent of the zero function in mathematics - indeed that's exactly what it is, if L is "number".) So:

```
let Q be a phrase numbers -> times;
showme Q;
showme Q applied to 4;
showme Q applied to -7;
```

produces:

```
"q" = phrase number -> time: default value of phrase number -> time
"q applied to 4" = time: 9:00 am
"q applied to -7" = time: 9:00 am
```

Here Q is set to the default phrase because we didn't give it any other value; it has the name we might expect ("default value of phrase number -> time") and it works as advertised, producing 9:00 am no matter what number is fed in.

More ambitiously, and supposing that we have a kind called "colour" whose first possible value is "red":

```
let R be a phrase numbers -> (phrase numbers -> colours);
showme R;
showme R applied to 3;
showme (R applied to 3) applied to 2;
```

produces:

```
"r" = phrase number -> (phrase number -> colour): default value of
phrase
number -> (phrase number -> colour)
"r applied to 3" = phrase number -> colour: default value of phrase
number
-> colour
"( r applied to 3 ) applied to 2" = colour: red
```

§22.5. Map, filter and reduce

When a mass of computations has to be done, the traditional approach is to work through them in a "repeat" loop. One modern alternative, growing in popularity, is to form a list of inputs; then apply the same computation to each input in turn to form a list of results (this is called "mapping"); throw out any bad or unwanted results ("filtering"); and then combine the surviving results into a single composite answer ("reducing", though some programming languages call this "folding" or "accumulation"; it's a much-reinvented idea).

Inform provides all three of these fundamental list-processing operations. There is no special term for a "map", because Inform treats it as another case of "applied to".

(phrase value -> value) **applied to** (list of values) **... value**

This phrase takes the list, applies the phrase to each entry in the list, and forms a new list of the result. Example:

```
To decide what number is double (N - a number) (this is
doubling):
    decide on N plus N.
```

Then "doubling applied to 2" produces 4, by the simpler definition of "applied to", but also:

```
doubling applied to {2, 3, 4}
```

produces the list {4, 6, 8}.

More divertingly, suppose we define:

```
To decide what text is the longhand form of (N - a number)
    (this is spelling out):
    decide on "[N in words]".


To decide what text is the consonant form of (T - text)
    (this is txtng):
    replace the regular expression "<aeiou>" in T with "";
    decide on T.
```

Then we can write a chain of three maps in succession:

```
txtng applied to spelling out applied to doubling applied to {3, 8, 4, 19,
7}
```

to produce the value {"sx", "sxtn", "ght", "thrty-ght", "frtn"}.

Next, filtering. Here we make use of descriptions, in order to say what values will be allowed through the filter. So:

filter to (description of values) **of** (list of values) **... value**

This phrase produces a new list which is a thinner version of the one given, so that it contains only those values which match the description given. Example:

```
filter to even numbers of {3, 8, 4, 19, 7}
```

produces {8, 4}, with the values 3, 19, and 7 failing to make it through. A sufficiently fine filter may well thin out a list to a single entry, or even no entries at all, but the result is always a list.

To get the full effect of filtering, we probably need to define an adjective or two. For example:

```
Definition: a text (called T) is lengthy if the number of characters in it
is greater than 6.
```

We can then write:

```
filter to lengthy texts of spelling out applied to {15, 2, 20, 29, -4}
```

which produces {"fifteen", "twenty-nine", "minus four"}.

Lastly, reduction. In order to combine a whole list of values, we need a phrase to combine any two. Here are some samples:

```
To decide what number is the larger of (N - number) and (M -
number)
    (this is maximizing):
    if N > M, decide on N;
    decide on M.
```

```
To decide what text is the concatenation of (X - text) and (Y - text)
    (this is concatenation):
    decide on "[X][Y]".
```

And here are some sample reductions:

```
let X be the maximization reduction of {3, 8, 4, 19, 7};
let Y be the concatenation reduction of txtng applied to spelling out
    applied to doubling applied to {3, 8, 4, 19, 7};
```

sets X to 19, the highest of the values, and Y to the text "sxsxtnghtthrty-ghtfrtn". In each case a list has been reduced to a single value which somehow combines the contents.

(phrase (value, value) -> value) **reduction of** (list of values) **... value**

This phrase works through the list and accumulates the values in it, using the phrase supplied. Example: if we have

```
To decide what number is the sum of (N - number) and (M -
number)
    (this is summing):
    decide on N + M.
```

then the summing reduction of {3, 8, 4, 19, 7} is the number 41, obtained by

```
(((3 + 8) + 4) + 19) + 7
```

so that the summing phrase has been used four times.

Is map/filter/reduce always a good idea? Devotees point out that almost any computation can be thought of in this way, and in systems where the work has to be distributed around multiple processors it can be a very powerful tool. (There are programming languages without loops where it's essentially the only tool.) At its best, it reads very elegantly: one assembles all of the tools needed - definitions of doubling, lengthy, spelling out, concatenation and so on - and then each actual task is expressed in a single line at the end.

On the other hand, there are also times when this is a needlessly complicated disguise for what could more easily be done with a "repeat" loop, and also more efficiently since assembling and dismantling lists in memory does take some overhead time. So these list operations are not a panacea, but it's good to have them available.

Start of Chapter 22: Advanced Phrases

Back to §22.4. Default values for phrase kinds

Onward to §22.6. Generic phrases

§22.6. Generic phrases

The following looks quite innocent:

```
To say (V - value) twice: say "[V]. [V], I say!"
```

It's clear at a glance what this is intended to do, but at a second glance things aren't so straightforward. "Value" is not itself a kind - it's too big and unspecific. For instance, if we were to allow a variable to be just "a value", we could freely set it to 12 one minute and to "dahlias" the next, and such a variable would be dangerous since we would never know what could safely be done with its contents. A phrase like this one is called "generic", because it's not so much a single, actual phrase as a recipe to make phrases. (Inform automatically works out which kinds we need the phrase for,

and creates a version of the phrase for those kinds.)

So "value" is not a kind, but a kind of kind. Inform has several of these:

```
value, arithmetic value, enumerated value, sayable value
```

These act as ways to say "a value of any kind matching this can go here". For example, "value" is a way to say "any kind at all"; "arithmetic value" is any kind which arithmetic can be performed on (any kind with the little calculator icon in the Arithmetic part of the Kinds index); and so on. If we write:

```
To double (V - arithmetic value): say "[V times 2]."
```

the restriction to "arithmetic value" means that although "double 3", "double 6 kg", etc., would be matched, "double the Entire Game" would not - you can't perform arithmetic on scenes. Similarly, it would have been tidier to write:

```
To say (V - sayable value) twice: say "[V]. [V], I say!"
```

because then Inform will make it clearer why "say X twice" won't work if X is one of those rare values which it can't say (an activity, for instance).

The Kinds index shows which kinds match against which of these "kinds of kind". For instance, it shows that "time"

```
Matches: value, arithmetic value, sayable value
```

which means that time is something we can do arithmetic on, and can say.

Start of Chapter 22: Advanced Phrases

Back to §22.5. Map, filter and reduce

Onward to §22.7. Kind variables

§22.7. Kind variables

The examples of generic phrases in the previous section were really only toy examples. Suppose we want a phrase which will take any arithmetic value and triple it. We could do something like this:

```
To triple (V - arithmetic value): say "[V times 3]."
```

But this only prints the answer. Suppose we want to be given the value back, instead: how can we write the phrase? The trouble is that, not knowing the kind of V, we can't say what kind will be produced. We need a way of saying "the same kind comes out as went in". Inform expresses that using kind variables:

```
To decide which K is triple (original - arithmetic value of kind K):
    decide on 3 times the original.
```

Here, K stands for any kind which matches "arithmetic value". Inform supports exactly 26 of these symbols, which are written A to Z, but it's customary to use K and L. (They can be written in the plural if we like: e.g., "list of Ks". But they must always use a capital letter: "list of k" is not allowed.)

Each symbol we use has to be declared in exactly one of the bracketed ingredients for the phrase - here, the declaration is "arithmetic value of kind K". That creates K and says that it has to be arithmetic; if we'd just said "value of kind K", it could have been anything. (Alternatively, we could use any of the kinds of kind in the previous section.)

For a more ambitious example, here is one way to define the mapping operation described earlier in the chapter:

```
To decide what list of L is (function - phrase K -> value of kind L)
    applied to (original list - list of values of kind K):
    let the result be a list of Ls;
    repeat with item running through the original list:
        let the mapped item be the function applied to the item;
        add the mapped item to the result;
    decide on the result.
```

Here we need two symbols to explain the complicated way that the values going in and out have to match up to each other. Note also the way that the temporary variable "result" is created:

```
let the result be a list of Ls;
```

Ordinarily, of course, "L" is not a kind. But within the body of a phrase definition, it means whatever kind L matched against.

When a symbol occurs several times in the same definition, subtle differences can arise according to which appearance is the declaration. These are not quite the same:

```
To hunt for (V - value of kind K) in (L - list of Ks): ...
To hunt for (V - K) in (L - list of values of kind K): ...
```

The difference arises - though very rarely - if V has some different kind compared to the list entries, but which can be used as if it were of that kind. For example,

```
hunt for the player's command in {"take all", "wait"};
```

Here V is a snippet, but L is a list of texts; and a snippet can be used in place of a text, but not vice versa. So this will match the second definition, because K is set to "text", but it won't match the first, where K is set to "snippet".

Start of Chapter 22: Advanced Phrases
Back to §22.6. Generic phrases
Onward to §22.8. Matching the names of kinds

§22.8. Matching the names of kinds

Sometimes a phrase needs to know what kind it's to work on, but isn't going to be given any particular value of it. For example:

```
To assay (name of kind of value K):
    repeat with item running through Ks:
        say "There's [item].";
    say "But the default is [default value of K].";
```

Note that there's no hyphen, and no name for the bracketed token - it only sets K. We can then have, say:

```
assay colours;
assay vehicles;
```

But "assay texts" would throw a problem message, because we can't repeat through all possible texts. For a different reason,

```
assay open doors;
```

would not be allowed - "open doors" is a description which applies to some doors and not others; it isn't a kind. It would make no sense to talk about "default value of open door", for example.

Start of Chapter 22: Advanced Phrases
Back to §22.7. Kind variables
Onward to §22.9. In what order?

§22.9. In what order?

Recall the definition:

```
To slam shut (box - an open container): say "With great panache, you
slam shut [the box].".
```

Suppose we then try to "slam shut the wall safe" at a time during play when the wall safe is already closed. An error message will then be printed during play, since there must be a mistake in the design. The combination of checking both when Inform builds the story file and then continuously when the story file is played guarantees that, in all cases, a varying item such as "box" in the definition of "To slam shut (box - open container)" always satisfies the condition laid down.

Instead suppose we also have the following definition:

```
To slam shut (box - a container): say "You are unable to slam shut
[the box], which is already closed.".
```

We now have two definitions of "slam shut". Sometimes the box it's applied to will be closed, in which case only the second definition fits, and will be the one used. But if the box is open, both definitions fit. Which will happen? The rule is:

1. A narrower condition takes precedence over a broader one;

2. If neither condition is narrower than the other, then whichever phrase was defined later in the source code is the one taking precedence;

3. Except that if the phrase is being used in the definition of phrase P, then P is always last in precedence order, so that recursion is always the very last possibility considered. This allows more specific or later definitions to make use of less specific or earlier ones in a natural way.

Rule 1 ensures that a definition involving "open container" takes priority over one which merely involves "container", for instance.

And therefore when the box is open, it's the more specific phrase to do with open containers which is invoked: so, with great panache, the box is slammed shut.

On the other hand, neither of these patterns is narrower than the other:

```
To describe (something - transparent): ...
To describe (something - container): ...
```

Some containers are transparent, some not; some transparent things are containers, some not. Rule 1 therefore does not apply, so it is the later of the two phrases which takes effect.

Start of Chapter 22: Advanced Phrases
Back to §22.8. Matching the names of kinds
Onward to §22.10. Ambiguities

§22.10. Ambiguities

Another possible ambiguity occurs when a phrase might match two lexically different

possibilities using the same words.

```
say the dishcloth;
```

could be construed as a usage of either of these cases:

```
say the (something - a thing)
say (something - a thing)
```

These of course have different effects - one produces the name with a definite article, the other just the name, so the difference is important.

The rule here is that whichever possibility contains the most words, in this case "say the (...)", takes precedence, because it's assumed to be a more specific form of the less wordy version.

Start of Chapter 22: Advanced Phrases
Back to §22.9. In what order?
Onward to Chapter 23: Figures, Sounds and Files: §23.1. Beyond text

---

Examples from Chapter 22: Advanced Phrases

Start of this chapter
Chapter 23: Figures, Sounds and Files
Indexes of the examples

439 **ExampleCurare** *RB*

A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

```
"Curare"


A thing has a number called the last use. The last use of
a thing is usually 0.


Definition: a thing is old if its last use is 12 or less.
```

The actual number chosen in this definition is pretty much irrelevant: the main thing is that we want to establish relative values. The lower the "last use" number of an item, the older that item should be understood to be, as we see here:

```
To decide which thing is cyclically random (collection - a
description of objects):
    let choice be the oldest member of the collection;
    now the last use of the choice is the turn count;
    decide on choice.
```

This phrase will select, from the collection of objects passed to it, the one that has been mentioned least recently. This means that if we consult it repeatedly about the same collection, it will begin to cycle predictably; but if new items are added to the collection, it will mention these first before returning to the previous cycle. Now we can use this:

```
After taking inventory:
    say "You stare morosely at [the cyclically random
thing carried by the player], wondering what you're ever
going to find to do with it."
```

We could have said "You stare morosely at [the oldest thing carried by the player]" here, but doing so would not have set the "last use" property correctly, so we would not get the cycling behavior that we're looking for.

```
The Evidence Room is a room. Some shelves are
scenery supporters in the Evidence Room. A box is a
kind of container which is open and not openable. On
the shelves is a box. It contains a deformed bullet and a
driver's license.
```

The player carries a steel fish hook, a Chinese passport, a tube of synthetic curare, and an envelope full of Euros.

Test me with "i / i / i / i / i / i / i / get all from box / i / i / i".

Chapter 23: Figures, Sounds and Files

nds; *§23.8. Declaring and playing back sounds*; *§23.9. Providing accessibility text*; *§23.10. Some techn*

Contents of *Writing with Inform*
Chapter 22: Advanced Phrases
Chapter 24: Testing and Debugging
Indexes of the examples

§23.1. Beyond text

In this chapter, we explore a number of ways to go beyond the traditional text-only, one-story-file-only model for IF.

These relatively exotic features are more demanding of the interpreter which a player uses than a plain text story file would be. They can only be used if the project is being compiled to the Glulx story file format (see the Settings panel for the project), and even then, the player will need to have a good Glulx interpreter - one which is reasonably up to date and well-written, that is - to be sure of everything working as intended.

Start of Chapter 23: Figures, Sounds and Files
Back to Chapter 22: Advanced Phrases: §22.10. Ambiguities
Onward to §23.2. How IF views pictures

---

§23.2. How IF views pictures

Looking around a bookshop, perhaps half of all the books published have illustrations. The proportion may be lower for novels, but if we count maps or other occasional diagrams, even the fiction section turns out to be surprisingly pictorial. Illustrations do not suit every book, but they are an option we would like to have available.

In the cultural history of IF, graphics in text adventures have sometimes been looked

at with suspicion. Mostly this is because attempts in the 1980s were not very successful, because computer graphics were so poor then (by modern standards). It may be that some people also felt that the takeover of computer games by graphical interfaces was the death knell of IF. But pictures are now rendered in superb quality by computers, and the death of IF turned out to be an exaggeration, so it is time to move on.

Whether to have illustrations ought to be an artistic choice, like whether to include a romantic sub-plot or how much of the back story is revealed. But there are practical considerations too. The most successful illustrated books are those whose pictures are well-chosen, have a sense of design to them, and above all are consistent. Consider how much worse off *Winnie the Pooh* would have been if a selection of random teddy-bear drawings had been used, instead of E. H. Shepherd's beautifully conceived world; or a cookery book in which the recipes are all photographed at different distances and light levels. IF writers may want to look for collaborators with a visual eye, just as most novelists do not draw their own illustrations.

Another consideration is that displaying images is more complicated for computers than displaying text. Not all devices can show pictures (consider handheld gadgets) and if they do, they may use different colour ranges or resolutions. So IF with pictures is always just a bit less portable than IF without, and because of that we must next look again at IF story file formats.

Start of Chapter 23: Figures, Sounds and Files
Back to §23.1. Beyond text
Onward to §23.3. Virtual machines and story file formats

§23.3. Virtual machines and story file formats

Back in Chapter 2, we saw that the Settings panel allows any given Inform project to be produced in either of two possible story file formats. Recall that story files are the released IF works: what the player sees. The source text, the Index, and so on are not part of this.

A story file is not like a word-processed document, or a photograph. There are many rival formats for these - for instance, an image on a web page might be in JPEG or PNG format, among many others - but basically they are simple things for the reader to look at, and see everything in one go. An IF story file is more complicated, because the "reader" reacts to it, types in to it, is surprised by it, never quite knows what might happen next.

A story file is in fact a computer program in its own right, but not a program like iTunes or Firefox which runs on a typical home or business computer. Instead it is a

program for an imaginary computer, called a "virtual machine" or "VM". This has a design ideally suited to IF, and it would be the perfect IF player's computer if only it actually existed. Because it doesn't, the player instead runs an "interpreter" program like Windows Glulxe or Zoom or Spatterlight - and this one is a program like iTunes or Firefox - and the interpreter acts as a middle-man. It creates a software version of the virtual machine, and then runs the IF story file on that VM. This sounds slow and impractical, but in fact it works well, and is also much safer since programs on the VM are not allowed to touch the real computer - so they cannot at all easily contain viruses or other malware. (In theory a malicious story file might try to exploit a bug in one of the various VM implementations in use, just as malicious image files have been used to attack bugs in web browsers, but this has never in practice happened. Nothing can be absolutely safe, but a story file belongs in the "mostly harmless" category of files - like images - rather than the "how far do you trust this person?" category - like programs.)

The different formats of story file are programs for different virtual machines. Just as Windows and Mac OS X offer basically similar services to the user but are very different in appearance and their workings, so the different VMs are quite different. Some can display pictures, others not.

Start of Chapter 23: Figures, Sounds and Files

Back to §23.2. How IF views pictures

Onward to §23.4. Gathering the figures

---

§23.4. Gathering the figures

Inform provides basic support for displaying pictures and leaves more exotic effects for Extensions to provide. But either way, for reasons explained in the previous section, **we can only have pictures if the Settings for the project are set to the Glulx story file format**.

Inform calls these pictures "figures", following the usual Inform analogy with books. We will think of our work of IF as being like a mostly textual book which in broken up with illustrations here and there - Figure 1, Figure 2, and so on. These might be used to mark each new chapter of the plot, or each new location: whatever the author would like. So the first thing we need to do is decide when pictures should appear.

The second thing to do is to get hold of the pictures we want to use. These might be photographs, or artwork, or diagrams: anything, really, but we will need them to be in either JPEG or PNG format. Inform does not itself try to be an image editor, or an artwork program - there are many such programs already which do these things much better than Inform could.

The pictures then need to be put in a special place where Inform can reach them. Suppose the Inform project is called Example.inform. Then we need to create a folder alongside it called "Example.materials", and create a further folder inside that called "Figures". The actual images go inside "Figures". So we might then have files like so:

```
Example.inform
Example.materials
    Figures
        Woodlands.png
        Blackberry.jpg
        Red Admiral Butterfly.png
```

The ".materials" folder for an Inform project will turn out to have many other uses in the chapter on Publishing, and will be explained further there.

Start of Chapter 23: Figures, Sounds and Files
Back to §23.3. Virtual machines and story file formats
Onward to §23.5. Declaring and previewing the figures

§23.5. Declaring and previewing the figures

Inside Inform, the source text for a project always tries to avoid talking about filenames - we need a better way to refer to the individual figures.

We do this by declaring each figure with a sentence like the following examples:

```
Figure of Woodlands is the file "Woodlands.png".
Figure 2 is the file "Red Admiral Butterfly.png".
```

Figure names can consist of any text provided that text starts with the word "Figure". So "Figure 3 - Woodlands", for instance, or even "Figure W" would have been just as good as "Figure of Woodlands". Books tend to number figures, but then, in a book the order in which they appear is known in advance - which might not be true in IF.

The file names must be exactly those used in the Figures folder. We need not declare every image kept there, but those we don't declare - remember Blackberry.jpg? - cannot be displayed.

We can preview the stock of figures by going to the table of figures in the Contents index for a project (once the project has been built, that is, so that its index is up to date). This preview shows thumbnail forms of the pictures, the names, the formats and the image sizes in pixels. A warning triangle is shown for any images in the wrong format, or which are missing from the Figures folder.

Start of Chapter 23: Figures, Sounds and Files

Back to §23.4. Gathering the figures

Onward to §23.6. Displaying the figures

§23.6. Displaying the figures

Inform's basic picture support simply allows figures to be shown at particular times. Once seen, they scroll away, just as text does once it has been printed. These pictures are really part of the stream of narrative. (If we would like icons or other images to be permanently present on screen, and divide the screen up in pictorial ways to achieve interesting layouts, we need to use special extensions to access Glulx's more exotic features.)

Displaying a picture is therefore like printing some text. Rather than

```
say "The woodlands stretch from here to the horizon.";
```

we would use:

display (figure name)

This phrase causes the figure to be displayed in a way visible to the player. If the option "one time only" is used, it will have no effect if the figure has been displayed before. Example:

```
display the Figure of Woodlands;
```

Once again, note that the "display" phrase does nothing unless the Settings for the project are set to the Glulx story file format. When a Glulx work is released as a blorb

(the default setting for the way releases occur: see the chapter on Publishing), all the images used are automatically included.

Start of Chapter 23: Figures, Sounds and Files
Back to §23.5. Declaring and previewing the figures
Onward to §23.7. Recorded sounds

§23.7. Recorded sounds

Inform also supports the playing back of recorded sounds, which might be anything from a three-second sound effect for a creaking door to an epic orchestral symphony. **Sound support is very newly added to the system and work is still in progress. In particular, sounds are not played by Inform for OS X (although it does produce valid blorbed Glulx story files), though they should be audible from within the Inform application for Windows.**

Once again, sound effects are supported by Inform 7 only on the Glulx platform, and even then we must be prepared for the fact that not all interpreters will be able to play them back. We must also bear in mind that a sound recording is a large pile of bits, and that adding any kind of sounds will greatly increase the size of the Blorb file for the released Glulx story file.

The sound files provided must have one of two formats: AIFF or Ogg Vorbis. **AIFF** is a traditional format in the recording industry, though it is more familiar to Mac OS X users than Windows users. It is uncompressed, giving what can be excellent audio quality, but at the cost of sometimes enormous file sizes - perhaps as much as 10 MB per minute, though this can be greatly reduced by lowering the sampling frequency, and halved again by dropping from stereo to mono.

Except for very short sound effects, we recommend using **Ogg Vorbis** instead. This is a compressed format whose file sizes will typically be more like 1 MB per minute. Inform uses Ogg Vorbis as the only format safe from licencing and patent disputes. (We would very much have liked to provide MP3 support, but this is no longer legally possible for free software.)

Support for Ogg Vorbis is not built in to either Windows or Mac OS X, and any sound recording you make will probably have to be made first to another format (perhaps AIFF or WAV), and then converted. See **www.vorbis.com** for encoding software which can convert from other sound formats to Vorbis.

Lastly, it must be remembered that recording industry bodies are very hostile to established copyright law covering fair use, parody, quotation of insubstantial

passages, etc., when it comes to mixing or using commercially released music. They are well-resourced and highly litigious. If you use sound effects not originated by yourself, you do so at your own risk, even if what you do is perfectly legal on any reading of the statutes.

Start of Chapter 23: Figures, Sounds and Files

Back to §23.6. Displaying the figures

Onward to §23.8. Declaring and playing back sounds

§23.8. Declaring and playing back sounds

Sound effects are accommodated on the same basis as illustrations. The relevant media files need to be placed in a subfolder of the project's ".materials" folder, but this time called Sounds rather than Figures, so for instance:

```
Example.inform
Example.materials
    Figures
        Woodlands.png
        Blackberry.jpg
        Red Admiral Butterfly.png
    Sounds
        Rustling leaves.ogg
```

Again, these must be declared in the source text:

```
Sound of rustling leaves is the file "Rustling leaves.ogg".
```

And they can be played using a special phrase:

play (sound name)

This phrase causes the sound effect to be played. If the option "one time only" is used, it will have no effect if the sound effect has been played before. Example:

```
play the sound of rustling leaves;
```

Start of Chapter 23: Figures, Sounds and Files
Back to §23.7. Recorded sounds
Onward to §23.9. Providing accessibility text

§23.9. Providing accessibility text

It's conventional for web pages to provide "alt-text" for significant images displayed, so that partially sighted or blind users can get an idea of what is being shown. Inform allows figures to be given these short descriptions like so:

```
Figure 2 is the file "butterfly.jpg" ("A red admiral butterfly.").
```

As we'll see, the same can be done for the cover image:

```
Release along with cover art ("A cathedral at sunset.").
```

And also for sounds:

```
Fugue is the file "Bach.ogg" ("A church organ playing a Bach
fugue.").
```

Start of Chapter 23: Figures, Sounds and Files
Back to §23.8. Declaring and playing back sounds
Onward to §23.10. Some technicalities about figures and sounds

§23.10. Some technicalities about figures and sounds

(i) Names for figures, such as "Figure of Woodlands", are values for a special kind of value called "figure name". This can in turn be used to define variables, properties and phrases:

```
The turn card image is a figure name that varies.
```

```
An Old Master is a kind of thing. An Old Master has a figure name
called appearance. Figure 1 is the file "Giaconda.jpg". The Mona Lisa
is an Old Master. The appearance of the Mona Lisa is Figure 1.
```

```
To place (F - a figure name) in the gallery: ...
```

(ii) Similarly, names for sound effects, such as "Sound of rustling leaves", are values for the kind of value "sound name".

(iii) In the released, blorbed-up Glulx file, figures and sound effects are internally given resource ID numbers which count upwards from 2 in order of their declaration. (Figure and sound numbers can thus be intermingled, if their declarations are.) Resource ID number 1 is reserved for the image of the cover art, if there is any. (See the chapter on Publishing.) To obtain these numbers, if we need them, we can use:

Glulx resource ID of (figure name) **... number**

This phrase produces the ID number used in the eventual Glulx file for the given figure.

Glulx resource ID of (sound name) **... number**

This phrase produces the ID number used in the eventual Glulx file for the given sound effect.

(iv) Glulx hackers may also like to know that whenever Inform 7 builds a project for Glulx, the Inform 6 code it generates always contains a full copy of John Cater's definitive header file "infglk.h".

Start of Chapter 23: Figures, Sounds and Files
Back to §23.9. Providing accessibility text
Onward to §23.11. Files

§23.11. Files

Once an Inform project is released, it is playable as a "story file", which is in effect a computer program for a specially IF-adapted design of computer. Story files run in what in computing is sometimes called a "sandbox", a kind of safe play area where it can be guaranteed that they cannot do any harm. This is good, because it means a story file can't be infected with viruses or other malware. If the project's Settings panel has the story file format set to the Z-machine, the story file is so thoroughly boxed in that it cannot even see the bigger computer beyond: it lives in a world of its own. But the Glulx format opens the door a crack, allowing the story file to read and write a small number of data files, which live in a single folder on the bigger computer's hard drive.

Why might we want this? Among the reasons are -

- to remember what has happened in previous attempts by the player;
- to store the player's preferences;
- in a two-part story, where each part is an independently released story file, to allow Part I to save some information about its ending which Part II could then pick up and make use of;
- to communicate with some external program, such as an Internet service.

Start of Chapter 23: Figures, Sounds and Files
Back to §23.10. Some technicalities about figures and sounds
Onward to §23.12. Declaring files

§23.12. Declaring files

Like figures and sounds, files must be declared before they can be used. For instance:

```
The File of Glaciers is called "ice".
```

This creates a new named constant "File of Glaciers" to refer to the file, throughout the source text. We use this name for it whether or not the actual disc file exists yet: it might be one that will only be created if something unusual happens in play, for instance.

Quoted filenames should contain only letters and digits, should be 23 characters or fewer, and should begin with a letter. (In particular they can contain no slashes or dots - no subfolders or extensions can be indicated.) The actual filename this translates to will vary from platform to platform, but "ice.glkdata" is typical, stored in some sensible folder.

Every file has an owner - not a person, but the project which normally writes to it. Inform assumes that the current project will be owning any file which it declares - the File of Glaciers, for instance. But we can optionally specify that it is owned by somebody else:

```
The file of Boundaries (owned by another project) is called "milnor".
The file of Spectral Sequences (owned by project "4122DDA8-
A153-46BC-8F57-42220F9D8795") is called "adams".
```

Inform uses ownership to make sure that we do not accidentally read in a file which has nothing to do with us, but merely happens to use the same name. Thus it is an error to read a file whose ownership does not agree with our declaration. Saying that a file is owned by "another project" allows us to read it whatever the owner is (so this can be used for files shared between multiple projects); specifying exactly where it needs to come from allows us to pass information from one project to another. Note that we identify projects using the IFID number - this can be found in the Contents index for a project, or by typing VERSION during play; see the chapter on Publishing for more about IFIDs.

Files are indexed in the Contents index, alongside figures and sound effects.

Two technicalities. First, constants such as "File of Glaciers" are of a kind of value called "external file" (compare "figure name" and "sound name"). Second, Inform's file-handling is provided for the Glulx machine, which in turn uses the Glk interface. This allows for either text or binary files. Inform's higher-level phrases to do with files, described in this chapter, all use text files, and all declared files are text files by default. But we can optionally add the keyword "binary" to declare a binary file, if needed:

```
The binary File of Glaciation Data is called "icedata".
```

Start of Chapter 23: Figures, Sounds and Files
Back to §23.11. Files
Onward to §23.13. Writing and reading tables to external files

§23.13. Writing and reading tables to external files

The main use for files is to store and retrieve data, and the most flexible form of data used by Inform is the Table, so facilities are provided which make it as easy as possible to write and read the contents of a table to files. If so, the file must contain just one single table: so to write multiple tables, we need to write multiple files, one for each.

To save the contents of a table to a file, we use the phrase:

write (external file) **from** (table name)

This phrase causes the entire contents of the given table to be written out to the given file. Note that files must have been declared, and must be referred to by their Inform names, not by textual filenames. Example:

```
write File of Glaciation Data from the Table of Antarctic
Reserves
```

Any blank rows in the table are automatically moved to the bottom, and only the non-blank rows are written.

To load a file back into a table,

read (external file) **into** (table name)

This phrase causes the entire contents of the given table to be read in from the given file. Note that files must have been declared, and must be referred

to by their Inform names, not by textual filenames. Example:

```
read File of Glaciation Data into the Table of Antarctic
Reserves
```

Any rows left spare at the foot of the table are automatically blanked. On the other hand if the file is too large to fit into the table - with too many columns or too many rows - a run-time problem is produced.

We can check if a file already exists using:

if (external file) **exists:**

This condition is true if the file-system used by the player appears to contain a file with the right name. For example, if we declared:

```
The binary File of Glaciation Data is called "icedata".
```

and then tested

```
if the File of Glaciation Data exists, ...
```

then Inform would search for a file called "icedata". (The arrangements for where this might be stored, and its filename extension, vary from platform to platform.)

One unfortunate restriction must be kept in mind. Some of what is stored in tables is solid information whose meaning never changes: the number 342, for instance, means the same to everyone. But other information depends entirely on the current location of certain structures in memory - for instance, a rule is internally referred to by its memory location. This potentially changes each time Go or Replay is clicked, and so it is not safe to pass it from one copy to another, or from one project to another. The only tables which Inform allows us to write into files are those containing "safe" data: numbers, units, times of day and kinds of value with named alternatives. Scenes, rules or rulebooks, in particular, are not allowed.

Start of Chapter 23: Figures, Sounds and Files

Back to §23.12. Declaring files

Onward to §23.14. Writing, reading and appending text to files

Example 440: ★ **Alien Invasion Part 23** Keeping a preference file that could be loaded by any game in a series.

Example 441: ★★ **Labyrinth of Ghosts** Remembering the fates of all previous explorers of the labyrinth.

Example 442: ★★★ **Rubies** A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

---

§23.14. Writing, reading and appending text to files

Text can also be saved to a file, and again all file-handling is automatic:

write (text) **to** (external file)

This phrase makes the given text become the entire contents of the named file. Note that files must have been declared, and must be referred to by their Inform names, not by textual filenames. Example:

```
write "Jackdaws love my big sphinx of quartz." to the file of
Abecedary Wisdom;
```

append (text) **to** (external file)

This phrase adds the given text to the end of the current contents of the

named file (creating it if it does not exist on disc). Note that files must have been declared, and must be referred to by their Inform names, not by textual filenames. Example:

```
append "Jinxed wizards pluck ivy from the big quilt." to the
file of Abecedary Wisdom;
```

The quoted text can, of course, contain substitutions, so can be long and complex if need be.

Text from a file is printed back with the text substitution:

say "[text of (external file)**]"**

This text expands to the contents of the named file. Note that files must have been declared, and must be referred to by their Inform names, not by textual filenames. Example:

```
"[text of the File of Abecedary Wisdom]"
```

To copy one file to another, for instance,

```
write "[text of the file of Abecedary Wisdom]" to the file of Secondary
Wisdom;
```

Start of Chapter 23: Figures, Sounds and Files
Back to §23.13. Writing and reading tables to external files
Onward to §23.15. Exchanging files with other programs

Example 443: ★ **The Fourth Body** Notebooks in which the player can record assorted notes throughout play.

Example 444: ★★ **The Fifth Body** An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

---

§23.15. Exchanging files with other programs

Provided we declare the files in the right way, it is easy for one project to read a file created by another project.

But if we want more rapid communication, between two projects which are each playing at the same time, we need to be more careful. What if project A tries to read the file at the same moment that project B is writing it?

To avoid this, we have a concept of files being "ready". A file is ready if it exists, and is completely written, and not in use elsewhere. We have already seen:

```
if the file of Invariants exists...
```

But now we want a stronger condition:

> **if ready to read** (external file)**:**
>
> This condition is true if the file exists and is marked as being ready to read; that is, it is not in a state where another program is currently writing it. Example:
>
> ```
> if ready to read the file of Invariants, ...
> ```

A file cannot be ready to read if it does not exist, so this is a stronger condition. If A and B are attempting communication in real time, both running at once, then Project A should check that an external file owned by B is ready before it tries to read it. Files can also be marked as ready or not ready, in effect claiming them, thus:

> **mark** (external file) **as ready to read**

This phrase marks that we have finished writing to the given file, so that any external program is welcome to read it now. Example:

```
mark the file of Invariants as ready to read;
```

mark (external file) **as not ready to read**

This phrase marks that we are about to start writing to the given file, so that any external program should wait until we're finished if it wants to read the file. Example:

```
mark the file of Invariants as not ready to read;
```

Possibilities really begin to open up when project A is our story file, but B is not another story file at all: it is some external program such as a Web service, say. (Of course this is harder to set up, since the player needs to have both A and B running at the same time, but for stories running on an Internet server this can all be made seamless.)

When Inform begins writing a table, or text, to a file, it initially marks the file as not ready: only when the table or text is completely written and the file about to close is the file marked as ready.

In order to write non-story-file programs as B, communicating with story files as A, we need to know the file format used by Inform. An Inform file is currently a Unix text file (with 10 as the line division character), encoded as ASCII Latin-1. (We would like to use Unicode at some point in the future, but the Glk and Glulx layers are still not fully converted to Unicode.) It opens with a single header line in the form:

```
* //IFID// leafname
```

The opening character is an asterisk if the file is currently ready, a hyphen if the file is currently not ready. The IFID between the slashes is the IFID number of the project which last wrote to the file. (Marking "ready" or "not ready" does not count as a write for this purpose.) If an external program wrote the file, it should call itself something which will not clash with any story file's IFID. The leafname is the filename text used inside the story file where the file was declared. For instance:

Start of Chapter 23: Figures, Sounds and Files
Back to §23.14. Writing, reading and appending text to files
Onward to Chapter 24: Testing and Debugging: §24.1. Checking against the Index

Example 445: ★★★ **Flathead News Network** Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

Examples from Chapter 23: Figures, Sounds and Files

Start of this chapter
Chapter 24: Testing and Debugging
Indexes of the examples

★ **ExampleAlien Invasion Part 23** *RB*

Keeping a preference file that could be loaded by any game in a series.

Suppose that we have a series of games each of which allows the player to select a puzzle difficulty level. When the player plays a new game in the series, we want him to start out by default with the same difficulty level he faced earlier on, so we store this information in a small preferences file, as follows:

```
"Alien Invasion Part 23"
```

A difficulty is a kind of value. The difficulties are easy, moderate, hard, and fiendish.

Understand "use [difficulty] puzzles" as selecting difficulty. Selecting difficulty is an action out of world, applying to one difficulty.

Carry out selecting difficulty:
    choose row 1 in the Table of Preference Settings;
    now challenge level entry is difficulty understood;
    say "Puzzles will be [challenge level entry] from now on."

The File of Preferences is called "prefs".

When play begins:
    if File of Preferences exists:
        read File of Preferences into the Table of Preference Settings;
        choose row 1 in the Table of Preference Settings;
        say "(The current puzzle difficulty is set to [challenge level entry].)"

Check quitting the game:
    write File of Preferences from the Table of Preference Settings.

Table of Preference Settings
challenge level
easy

The Sewer Junction is a room.

Our preference file is restricted to a single option here for simplicity's sake, but we could keep track of more information -- whether the player preferred verbose or brief room descriptions, screen configurations, and so on.

If we were disposed to be somewhat crueler, we could use a similar method to make the player finish each episode of the series in order to "unlock" the next. All we would need to do is store a numerical password in our preferences file when the player finished a given level; the next level would check, say, the Table of Completed Levels for that password, and refuse to play unless the right number were present.

---

441 **Example** **Labyrinth of Ghosts** *RB*

Remembering the fates of all previous explorers of the labyrinth.

A tradition among Nethack-like computer games of the old school is that a player's death in a given place leaves a ghost behind to haunt subsequent players. Information about past lives is sometimes stored in a "bones file", and in this example we do exactly that, for a grievously unfair little dungeon.

To begin with, the labyrinth itself. We create a kind of value to remember possible means of death in these tunnels, and we assign a coordinate position in some grid to each location. (We do this because grid positions can safely be stored in tables saved out to external files, whereas room names cannot - they represent data which changes each time we amend the source.)

```
"Labyrinth of Ghosts"

Use scoring.

A demise is a kind of value. The demises are drowned,
buried by a rockfall, pierced by an arrow and slain. The
latest demise is a demise that varies.
```

A grid location is a kind of value. (1,19) specifies a grid location. A room has a grid location called coordinates.

The Gateway is a room. "For the foolhardy adventurer, the perilous labyrinth lies north, east or south." The coordinates are (6,6). The Tomb is east of the Gateway. The coordinates are (7,6). The Rockfall Cave is north of the Gateway. "This partly fallen cave may perhaps extend further north." The coordinates are (6,5). Instead of going north in the Rockfall Cave, have the player buried by a rockfall. The Archery Canyon is south of the Gateway. "No telling why this canyon is named after archery, but perhaps if you wait around you'll find out." The coordinates are (6,7). Instead of waiting in the Archery Canyon, have the player pierced by an arrow. The Rock Pool is east of the Tomb. The coordinates are (8,6). The cold mountain pool is in the Rock Pool. The cold pool is fixed in place. Instead of entering the cold mountain pool, have the player drowned.

Every turn when a random chance of 1 in 10 succeeds:
    say "A dwarf appears out of nowhere, and throws a nasty little knife.";
    have the player slain.

And as compensation for these hazards:

Some silver bars are in the Tomb. The emerald is in the Rock Pool. The platinum pyramid is in the Canyon.

Table of Point Values
item score

silver bars 8
platinum pyramid 12
emerald 4

Report taking an item listed in the Table of Point Values:

```
        increase the score by the score entry;
        blank out the whole row.
```

We are now ready for the actual undertaking. The Table of Ghostly Presences holds up to twenty death notices, and is initially blank. Deaths are sequentially numbered, and this number is stored in the sequence column.

```
Table of Ghostly Presences

[illegible]
a [illegible]
with 19 blank rows.
```

As the story file starts up, we look to see if a ghosts file already exists. If one does, we load up the Table of Ghostly Presences with it: and if not, as will be the case the first time the player explores, we leave the table blank. We sort the table so that it has earlier deaths (lower sequence numbers) first.

```
The File of Ghosts is called "ghosts".


When play begins:
    if the File of Ghosts exists, read File of Ghosts into
the Table of Ghostly Presences;
    sort the Table of Ghostly Presences in sequence
order.
```

How will ghosts manifest themselves? Because this is only a small example, we will simply tell the player that he senses something. If several ghosts are present in the same place, the most aggrieved (that is, the most recent) is sensed first...

```
After looking:
    repeat through the Table of Ghostly Presences in
reverse sequence order:
        if the haunted position entry is the coordinates of
the location, say "You sense the ghostly presence of an
adventurer, [manner of death entry] with a score of
[score at death entry] in [turns at death entry] turns."
```

(For instance, "You sense the ghostly presence of an adventurer, buried by a rockfall with a score of 10 in 5 turns.") That just leaves the rule for bumping off the player. When the Table is full, and there are already 20 ghosts, the one who died longest ago (with the lowest sequence count) is eliminated, and his row blanked out. (This will always be row 1 since we sorted the table in sequence order on reading it in.)

```
To have the player (sticky end - a demise):
    let the new sequence number be 0;
    repeat through the Table of Ghostly Presences:
        let S be the sequence entry;
        if S is greater than the new sequence number, let
the new sequence number be S;
    increment the new sequence number;
    if the number of blank rows in the Table of Ghostly
Presences is 0:
        choose row 1 in the Table of Ghostly Presences;
        blank out the whole row;
    choose a blank row in the Table of Ghostly
Presences;
    now the sequence entry is the new sequence
number;
    now the manner of death entry is the sticky end;
    now the turns at death entry is the turn count;
    now the score at death entry is the score;
    now the haunted position entry is the coordinates of
the location;
    write the File of Ghosts from the Table of Ghostly
Presences;
    now the latest demise is the sticky end;
    end the story saying "You have been [latest demise]".
```

Strictly speaking we ought to worry that after 2,147,483,647 deaths, the sequence numbers would grow too large to store in a single value, and then the sequence of ghosts will be erratic. But it seems unlikely that anyone will play this example 2.1 billion times.

---

442 ExampleRubies

RB

A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

The trick here is that we need a table with text in order to keep track of the players' names.

Part 1 largely replicates the source from "Identity Theft"; new material starts at Part 2.

"Rubies"

Use scoring.

Part 1 - Collecting Names

The player's forename is a text that varies. The player's full name is a text that varies.

When play begins:
    now the command prompt is "What is your name? > ".

To decide whether collecting names:
    if the command prompt is "What is your name? > ", yes;
    no.

After reading a command when collecting names:
    if the number of words in the player's command is greater than 5:
        say "[paragraph break]Who are you, a member of the British royal family? No one has that many names. Let's try this again.";
        reject the player's command;
    now the player's full name is the player's command;
    now the player's forename is word number 1 in the player's command;
    now the command prompt is ">";
    say "Hi, [player's forename]!";
    say "[banner text]";

```
    move the player to the location;
    reject the player's command.
```

Instead of looking when collecting names: do nothing.

Rule for printing the banner text when collecting names: do nothing.

Rule for constructing the status line when collecting names: do nothing.

Part 2 - Adding the Leaderboard

File of Leaderboard is called "leaderboard".

```
When play begins:
    if the File of Leaderboard exists:
        read File of Leaderboard into the Table of Leaders;
        sort the Table of Leaders in reverse scored amount
order.
```

```
When play ends:
    choose row 10 in the Table of Leaders; [we've sorted
the table, so the lowest score will be the one at the
bottom]
    if the score is greater than scored amount entry:
        now name entry is the player's forename;
        now the scored amount entry is the score;
    show leaderboard;
    write the File of Leaderboard from the Table of
Leaders.
```

```
To show leaderboard:
    sort the Table of Leaders in reverse scored amount
order;
    say "Current leading scores: [paragraph break]";
    say fixed letter spacing;
    repeat through Table of Leaders:
        if scored amount entry is greater than 0:
            say " [name entry]";
            let N be 25 minus the number of characters in
name entry; [here we want to space out the scores so
they make a neat column]
            if N is less than 1, now N is 1;
            say N spaces;
            say "[scored amount entry][line break]";
    say variable letter spacing.



To say (N - a number) spaces:
    repeat with index running from 1 to N:
        say " ".



Table of Leaders

name                scored amount
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
```

And now we introduce a scenario that allows different players to come up with different scores -- admittedly not a very interesting scenario, but it will do for now:

```
Part 3 - Scenario
```

```
Carry out taking something which is not handled:
    increment score.


The Big Treasure Chamber is a room. It contains a ruby,
an emerald, a gold tooth, an antique katana, and a silver
coin.


Instead of waiting, end the story finally.


Test me with "get ruby / z".
```

---

443 **Example** **The Fourth Body** *RB*

Notebooks in which the player can record assorted notes throughout play.

Some mystery games supply the player with an in-game system for taking notes, in case he doesn't want to rely on scraps of paper next to the computer. One way of doing this is to write out all the player's notes and observations into a notebook file, whose contents can be retrieved during play (or, indeed, after it).

We'll first invent a general system for writing text into notebooks, by creating a new kind called jotter. Each individual jotter will have its own disc file, and there will be basically three things which can be done with jotters: erasing, reading and writing.

```
"The Fourth Body"


A jotter is a kind of thing. A jotter has an external file
called the text file. A jotter can be fresh or used. A jotter
is usually fresh. A jotter has a text called the heading.
```

```
The currently erased jotter is an object that varies.
To erase (pad - a jotter):
    now the currently erased jotter is the pad;
    write "[heading of the currently erased jotter]
[paragraph break]" to the text file of the pad;
    now the pad is fresh.



To write in (pad - a jotter):
    append "[the time of day]: [topic understood][line
break]" to the text file of the pad;
    now the pad is used.



To read (pad - a jotter):
    say "You read: [paragraph break][text of the text file
of the pad]".
```

This is all as might be expected, except perhaps for the business of the "currently erased jotter". Why copy "pad" into this - why not simply write "[heading of the pad]"? The answer is that "pad" is a temporary "let" value, and cannot be used inside other phrases, such as the "write ... to ..." phrase.

We want to erase any jotters when play begins, as otherwise text left over from any previous games will still be visible:

```
When play begins:
    repeat with pad running through jotters:
        erase the pad.
```

Now we need to create rules to allow the player to control reading, writing and erasing. Reading we will handle with the ordinary examining action, but we create new actions for writing and erasing. A nice little trick allows WRITE WHATEVER to default to writing WHATEVER in a notebook being carried.

```
Instead of examining a used jotter (called the pad):
    read the pad.



Instead of examining a fresh jotter (called the pad):
```

```
    say "There is nothing of note in [the pad]."
```

```
Understand "write [text] in [something preferably held]"
as writing it in. Understand "write [text]" as writing it in.
Writing it in is an action applying to a topic and one
thing. Rule for supplying a missing second noun while
writing: if a jotter (called the pad) is carried, now the
second noun is the pad; otherwise say "You will have to
specify what to write that it."
```

```
Check writing it in:
    if the second noun is not a jotter, say "It would be
better to write in a notebook." instead.
```

```
Carry out writing it in:
    write in the second noun.
```

```
Report writing it in:
    say "Under the current time, you write '[the topic
understood]' into [the second noun]."
```

```
Understand "erase [something preferably held]" as
erasing. Erasing is an action applying to one carried
thing.
```

```
Check erasing:
    if the noun is not a jotter, say "It's hard to see how."
instead.
```

```
Carry out erasing:
    erase the noun.
```

```
Report erasing:
    say "You scrub out all the entries in [the noun]."
```

That completes a general-purpose implementation of jotters, and we put it to use:

```
The player carries a jotter called your notebook. The file
of Player's Observations is called "notebook". The text
file of your notebook is the file of Player's Observations.
The heading of your notebook is "Observations in the
Pottingham Green Case".


The Damp Hillside is a room. "It is just after dawn:
among the trees there is misty and pale blue light. [if
Havers is in the location]The only saturated color in view
is the orange-and-yellow jacket of [Detective Havers].
She is trying unsuccessfully to light a cigarette. [end if]
[paragraph break]The body itself is further down, closer
to the bottom of the ravine. It would be foolish to
speculate before seeing it, but the odds are that the
corpse was rolled down after death. The ground is not
steep enough for the fall itself to be deadly."


Detective Havers is a woman in the Damp Hillside. The
description is "She gives you a weak smile when you
look at her: you know she hasn't slept more than three
hours any of the last few nights." Havers is scenery.


Havers is carrying a jotter called Barbara's notebook.
The file of Barbara's Observations is called "barbara".
The text file of Barbara's notebook is the file of
Barbara's Observations. The heading of Barbara's
notebook is "I could murder a cup of tea".


The time of day is 6:32 AM.
```

Instead of examining your notebook when your notebook is fresh:
    say "Your notebook is blank. Back in the office, of course, there are a stack of others. But you brought a fresh notebook in a kind of weary hope. You're going to pretend, just for now, that this body might be unrelated to the graphic string of murders you're already investigating."

---

444 **Example** **The Fifth Body** *RB*

An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

The implementation here is much like that of the previous example, except that we allow the player to write his notebook input as a separate command, leading to an exchange such as

>write in my notebook
You open your notebook and prepare to write in it.

>>Am beginning to think that HT and BGG are in this together.
You finish writing and fold your notebook away.

>read my notebook
You read:

Wednesday morning

Am beginning to think that HT and BGG are in this together.

The opening is much as before:

"The Fifth Body"

A jotter is a kind of thing. A jotter has an external file called the text file. A jotter can be fresh or used. A jotter is usually fresh. A jotter has a text called the heading.

The currently erased jotter is an object that varies.

To erase (pad - a jotter):
    now the currently erased jotter is the pad;
    write "[heading of the currently erased jotter][paragraph break]" to the text file of the pad;
    now the pad is fresh.

To write in (pad - a jotter):
    append "[the time of day]: [player's command][line break]" to the text file of the pad;
    now the pad is used.

To read (pad - a jotter):
    say "You read: [paragraph break][text of the text file of the pad]".

When play begins:
    repeat with pad running through jotters:
        erase the pad.

Instead of examining a used jotter (called the pad):
    read the pad.

```
Instead of examining a fresh jotter (called the pad):
    say "There is nothing of note in [the pad]."


Target jotter is an object that varies. The target jotter is
usually nothing.


Understand "write in [something preferably held]" as
writing in. Writing in is an action applying to one thing.


Check writing in:
    if the noun is not a jotter, say "It would be better to
write in a notebook." instead.


Carry out writing in:
    now the command prompt is ">>";
    now the target jotter is the noun.


Report writing in:
    say "You open [the noun] and prepare to write in it."
```

Now what happens is that the player, having typed WRITE IN NOTEBOOK, will be faced with a ">>" prompt instead of the usual ">": a sign that the input mode has changed.

The next code is to react to reading a command. Whatever the player types at the >> prompt when the target jotter is set will now be recorded in the notebook, though with a character limit of about 60-100 characters depending on how much upper-case and punctuation he uses. (There are ways to lift the character length restriction as well, but they would take us into deeper waters.)

```
After reading a command when target jotter is a jotter:
    now the command prompt is ">";
    write in target jotter;
    now target jotter is used;
    say "You finish writing and fold your notebook away.";
```

now the target jotter is nothing;
reject the player's command.

Understand "erase [something preferably held]" as erasing. Erasing is an action applying to one carried thing.

Check erasing:
if the noun is not a jotter, say "It's hard to see how." instead.

Carry out erasing:
erase the noun.

Report erasing:
say "You scrub out all the entries in [the noun]."

The player carries a jotter called your notebook. The file of Player's Observations is called "notebook". The text file of your notebook is the file of Player's Observations. The heading of your notebook is "Sunday Morning".

The Vestry is a room. "[Havers] hangs back by the door: the forensics expert is not finished with a preliminary examination of the body. From here you can't see much, except that the expert has peeled back and laid to one side a liturgical vestment that someone at the church used to cover the corpse until the police came. What was once a cream silk with festive Easter embroidery is now stained with blood-colored handprints."

Detective Havers is a woman in the Vestry. The description is "She looks glumly back. There's still a purple-ish bruise on her cheekbone from the disaster

Thursday afternoon." Havers is scenery.

Havers is carrying a jotter called Barbara's notebook. The file of Barbara's Observations is called "barbara". The text file of Barbara's notebook is the file of Barbara's Observations. The heading of Barbara's notebook is "Sun. AM".

The time of day is 9:11 AM.

---

445 **ExampleFlathead News Network** *RB*

Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

This example can only work if we have a separate program running in the background while the story file is being played, and as such it will only work if we set things up in a special way. Exactly how to do this will vary from platform to platform.

First, the source text for the Inform end of the communication line:

"Flathead News Network"

The file of RSS Requests is called "rssrequest".

The file of RSS Responses (owned by project "RSS-SCRIPT") is called "rssreply".

To request (RSS feed address - text):
    mark the file of RSS Responses as not ready to read;
    write the RSS feed address to the file of RSS

```
Requests.


Newsroom is a room. "This is the secret nerve-centre of
FNN, the Flathead News Network."


The BBC button is in the Newsroom. Instead of pushing
the BBC button: say "Bong!"; request
"newsrss.bbc.co.uk/rss/newsonline_uk_edition/
front_page/rss.xml".


The NASA button is in the Newsroom. Instead of
pushing the NASA button: say "Bang!"; request
"www.nasa.gov/rss/breaking_news.rss".


The WHO button is in the Newsroom. Instead of
pushing the WHO button: say "Bing!"; request "http://
www.doctorwhonews.net".


A screen is in the Newsroom.


Instead of examining the screen:
    if ready to read the file of RSS Responses, say "From
the screen you read:[line break][text of the file of RSS
Responses]";
    otherwise say "The screen remains blank for now."
```

As far as the story file is concerned, then, it sends a request down the communication line by writing the chosen RSS feed to the file named "rssrequest", and expects a reply to come back down the line by being written to the file "rssreply". However, the story file needs to expect that this might take some time. (Maybe forever, if there is no program responding, or if the Internet connection is not working.) The story file marks the "rssreply" file as not ready before it makes a request; if it subsequently finds that the file is now ready, that must mean that the other program has done the honours, and that all is well. In the mean time, "The screen remains blank for now."

Now for the RSS-SCRIPT program. The following provides a crude but workable program suitable for running as a Perl script on a system which provides the standard Internet fetching program "curl": Mac OS X, for instance. (If you have OS X, you can paste the following into a (Unix-format) text file called "rss-script.pl", place it in your home folder, open the Terminal utility, type "perl rss-script.pl", and then hide the Terminal window again.)

```
for (;;) { # repeat forever:
   system("sleep 1"); # wait 1 second
   open REQUEST, "rssrequest.glkdata" or next;
   # the request file has been detected:
   $header_line = <REQUEST>; # the header line
   $rss_feed = <REQUEST>; # the actual content - the
RSS feed URL
   close REQUEST;
   if ($header_line =~ m/^\*/) { # if the request file is
marked ready
      $rss = system("curl $rss_feed >rawrss.txt"); #
download the RSS feed
      # read the RSS XML into a single Perl string:
      open RAWRSS, "rawrss.txt" or next;
      $raw = "";
      while ($nl = <RAWRSS>) {
         $raw = $raw." ".$nl;
      }
      close RAWRSS;
      # look for the title and description of the first item:
      if ($raw =~ m/\<item\>\<title.*?\>(.*?)\<\/title\>.*?
\<description.*?\>(.*?)\<\/description\>/) {
         # write the reply:
         open REPLY, ">rssreply.glkdata" or next;
         print REPLY "* //RSS-SCRIPT// rssreply\n", $1,
"\n", $2, "\n";
         close REPLY;
         # request safely dealt with, so we can remove it:
         system("rm 'rssrequest.glkdata'");
      }
   }
}
```

Chapter 24: Testing and Debugging

mmands; *§24.4. Low-level debugging commands*; *§24.5. Adding new testing verbs and Release for Tes*

Contents of *Writing with Inform*

Chapter 23: Figures, Sounds and Files

Chapter 25: Releasing

Indexes of the examples

§24.1. Checking against the Index

Testing a story -- and indeed writing a story so that it is easy to test consistently -- is an art in itself. We should expect that we'll do some preliminary testing, both by running test commands and by playing through the story ourselves, and that we'll then hand on the story to beta-testers who will tell us about faults in the play experience that we haven't been able to see.

Every time Inform builds a new story file, it assembles a vast amount of information about that world, in the form of the Index. Often a visit to the Index is all that's needed to explain a piece of undesired behavior.

Is travel not working as it should? Check the World index and see whether the map shows the rooms arranged the way you thought.

Are objects not showing the behavior you'd expect based on their kind? Check the Kinds index and make sure they've been defined as the kind of thing you expected. For instance, we might find that we've written

```
The red door is west of Foo and east of Bar.
```

but not

```
The red door is a door.
```

A human reader wouldn't make this mistake, but Inform hasn't actually registered the red door as belonging to the door kind, and consequently has treated it as a room instead. All we need to do is add the kind declaration. The Kinds index will make that

obvious.

When an error appears in the Index, there is often a link back to the source text that defined that room or object. If not, there's often at at least some information about what rule or phrase might be responsible for it.

Start of Chapter 24: Testing and Debugging

Back to Chapter 23: Figures, Sounds and Files: §23.15. Exchanging files with other programs

Onward to §24.2. Debugging features to use in source

§24.2. Debugging features to use in source

The TEST command is an extremely useful way of managing a story and continuing to verify that it does everything we want. We can create new test commands of the form

```
Test me with "up / kill captain eo".
Test eo with "zap eo" holding the ray gun.
Test dinner with "eat bread / eat soup / eat butter" in the Ship Cafeteria.
```

and we are free to have as many of these tests as we would like. Test commands can call other tests, as well, so we might have a test command such as

```
Test megatest with "test me / test eo".
```

A word of warning: if the first command in the test is "again", that will likely repeat the TEST command, sending Inform round in circles forever.

For complicated objects and commands, sometimes it's a good idea to develop the test commands at the same time that we're writing the source code itself. Each time we add a new rule or piece of behavior, we also add to that object's special test command something that will put that new feature to the test. This means that we can keep running the test command as we work and verify that everything is behaving as expected.

Sometimes we need to get a look at what is happening within the source itself. Many of the most annoying bugs come about because we're making some assumptions about what's true in the story world that differ from Inform's assumptions. When that happens, we may need to add something to the source to check that the variables are set to what we think, that certain parts of the source are being reached, and so on.

For instance, suppose we have a phrase like this:

```
To say key score:
    let count be the number of keys which are not carried by the
player;
    if count is greater than 2 and the player is timid:
        say "You're still missing a lot of keys, bucko!"
```

Now, we expect this to print something, but perhaps it's not doing so when we had anticipated that it would. At some point when we think the count is greater than 2 and the player is timid, at least one of those things is not true. An easy way to check up on this is to add a showme line to the source, like so:

```
To say key score:
    let count be the number of keys which are not carried by the
player;
    showme count;
    if count is greater than 2 and the player is timid:
        say "You're still missing a lot of keys, bucko!"
```

and this will then check the relevant number and print it to screen when this phrase is called, like so

```
"count" = number: 1
```

In this case, it looks like the count is not high enough to trigger the text, so we can concentrate on working out why that might be. Maybe we didn't correctly define something as a key, for instance.

Start of Chapter 24: Testing and Debugging
Back to §24.1. Checking against the Index
Onward to §24.3. High-level

§24.3. High-level debugging commands

If an object is not responding in the way we expect, it may be that we're wrong about where it is or about some of its current properties or relations. We can find our current location and the things around us by typing

```
>SHOWME
Boudoir - room
    four-poster bed - supporter
    yourself - person
    pillow
```

and similarly we can inquire about the status of a particular object during play by typing SHOWME and the object's name:

```
>SHOWME BAT
bat - thing
location: on the table in Locker Room
singular-named, improper-named; unlit, inedible, portable, patterned
printed name: "bat"
printed plural name: none
indefinite article: none
description: none
initial appearance: none
```

This will work even if we're not in the same location as the object we want shown.

Another common type of problem is one in which we type a command but Inform does not perform the action that we were expecting as a result. In some cases, this is because the command we're typing is actually triggering some other action. An easy way to check on this is to type ACTIONS before issuing the command that is behaving unsatisfactorily. Thus:

```
>ACTIONS
Actions listing on.
```

```
>JUMP
```

```
[jumping]
You jump on the spot.
[jumping - succeeded]
```

This tells us how Inform interpreted our input and whether the action was successful or failed for some reason. If the command is being understood as a different command than we expected, that may mean that we have made a mistake in our Understand instructions, and need to double-check these.

Sometimes, however, the action is being correctly understood, but the action rules that are firing are producing a result other than we'd like. If we want to see which rules are running, we can type

```
>RULES
Rules tracing now switched on. Type "rules off" to switch it off again,
or "rules all" to include even rules which do not apply.
```

```
>JUMP
[Rule "announce items from multiple object lists rule" applies.]
[Rule "set pronouns from items from multiple object lists rule"
applies.]
[Rule "before stage rule" applies.]
[Rule "instead stage rule" applies.]
[Rule "investigate player's awareness before action rule" applies.]
[Rule "player aware of his own actions rule" applies.]
[Rule "check stage rule" applies.]
[Rule "carry out stage rule" applies.]
[Rule "after stage rule" applies.]
[Rule "investigate player's awareness after action rule" applies.]
[Rule "report stage rule" applies.]
[Rule "report jumping rule" applies.]
You jump on the spot.
[Rule "last specific action-processing rule" applies.]
```

```
[Rule "A first turn sequence rule" applies.]
[Rule "every turn stage rule" applies.]
[Rule "A last turn sequence rule" applies.]
[Rule "notify score changes rule" applies.]
>
```

As we can see, RULES produces a lot of output, much of which is probably irrelevant to whatever problem we're tracking down. Nonetheless, knowing exactly which rule is printing undesirable output is helpful, especially if that rule comes out of an extension

or some other source that we did not write ourselves: this output has told us that the text we saw came from the "report jumping rule".

To find out more about what is going on in specific rules, we can also turn to the Index tab under Actions and click through to that specific action. From there we will be able to see which rules are included, what responses they're writing, and where they were defined in the source text.

SCENES lists which scenes are currently playing and which are complete. This is valuable if scene-triggered events are not happening when we expect them to.

RANDOM sets the random number generator to a predictable seed. If we include this in a test command, it will guarantee that the subsequent behavior of the story is consistent across multiple playthroughs, which is helpful if we're trying to test something to do with, say, randomly wandering non-player characters.

RELATIONS lists all the relations defined in the story, except for things like support and containment that are part of the world model and are so numerous that the output would be overwhelming.

RESPONSES lists all the named responses established by all the extensions currently included. This can be informative, or it can be a bit overwhelming. Except where responses have been changed at runtime, the same information is available in a different form in the Index on Actions. If we're interested in a particular single response, digging into the actions index is probably the easiest way to find it.

If, however, we want a rapid overview of all the responses provided by a given extension (perhaps an extension we are ourselves writing), the RESPONSES command can be a help.

Start of Chapter 24: Testing and Debugging
Back to §24.2. Debugging features to use in source
Onward to §24.4. Low-level debugging commands

§24.4. Low-level debugging commands

In addition to the commands designed around Inform 7's data model, there are several debugging commands that have persisted since the days of Inform 6. These are generally less useful or necessary with Inform 7's features, but there are still times when they can come in handy for a quick and dirty resolution of a problem during gameplay. They are as follow.

PURLOIN moves an object to your possession, no matter where it is on the map, like

so:

```
>PURLOIN TABLE
[Purloined.]


>I
You are carrying:
a table
```

Note that purloin does not consider the usual rules about whether something can be taken. In this case, we've just moved the table to our inventory even though it is a fixed in place supporter that could not be taken in the normal course of events.

Because purloin works on things that are far away as well as things that are close, it has to do a lot of extra parsing work and may take a moment or two to complete if we try it in a very large story. It is generally more efficient to give the player the relevant object using a testing command, like this:

```
Test me with "drop table" holding the table.
```

Nonetheless, there are occasionally times when we're halfway into a 2000-move story and suddenly realize we implemented a vital object in the wrong room, making the story unsolvable. We could fix the bug, press replay and return to this story state fairly quickly, but if we don't feel like waiting even that long, PURLOIN will resolve the issue.

ABSTRACT is PURLOIN's less useful cousin, allowing the player to move an object from one place to a specified other place, as in

```
Bar
You can see a table here.


>ABSTRACT KEY TO TABLE
[Abstracted.]


>LOOK
```

```
Bar
You can see a table (on which is a key) here.
```

GONEAR transports the player instantly to the vicinity of the named object, so for instance

```
>GONEAR GRAIN
Fertile Plain
You can see some grain here.
```

As a debugging command, this isn't protected in the ways that commands usually are. It's possible to type GONEAR NORTH and produce a run-time error when Inform tries to move the player into the object that represents the compass. Again, except in cases where we're tracing a problem very deep in an already running story, it is usually more practical to write a test command to put the player in the correct situation, as in

```
Test me with "eat grain" in the Fertile Plain.
```

VERIFY checks that the story file is intact rather than damaged, but it is hard to think of an occasion when this would be likely to arise within the Inform application. The command is a holdover from a time when data transfer was much slower and more error-prone, and it was plausible to have a story file of just a few hundred KB corrupted during transmission.

TREE creates a list of object containment. It is similar to SHOWME, but less elegant and thorough.

SCOPE lists the objects that are currently in scope for the player, which is to say, things that could be referred to when we're typing a typical command. Thus:

```
Bar
You can see a table here.
```

```
>SCOPE
1: yourself (574631)
2: a table (574759)
```

The following numbers are object IDs for these objects, which can distinguish items with identical names. It is likely that the output of this will not be terribly interesting or different from checking SHOWME, except in cases where the author is deliberately

changing the scope to be something other than "the set of things that are visible in the room with the player right now". This usually involves the Deciding the scope of something activity (see the chapter on Activities).

SHOWHEAP shows how many bytes are currently free. This is usually not helpful.

SHOWVERB (verbname) lists the Understand information associated with a particular verb. Similar information, in a vastly more palatable form, is available in Index / Actions / Commands, so the one time SHOWVERB becomes useful is when Inform is considering the understand lines in the wrong order and producing a result we didn't want: SHOWVERB will show us the order in which the lines are being assessed. The challenge will then be to add conditions to the Understand lines to move them into the correct order.

Finally, TRACE (and its more advanced stages TRACE 2, TRACE 3, TRACE 4, and TRACE 5) will reveal things, more things than we ever wanted to know, about the assumptions being made by the parser when it takes in a command. In practice this information is almost never useful to an Inform 7 author.

There is no guarantee that any of these commands will make life better or that they won't crash the story or put it into an unwinnable state. There is also no absolute guarantee that they won't be withdrawn entirely from future versions of Inform. Consider them as Old High Magic, and treat accordingly.

Start of Chapter 24: Testing and Debugging
Back to §24.3. High-level debugging commands
Onward to §24.5. Adding new testing verbs and Release for Testing

---

§24.5. Adding new testing verbs and Release for Testing

As we saw in Chapter 2, we can mark some of our source text so that it will not be included in a finished story. This means that we can add special testing commands available to the author but not available to our eventual players. This is a good way to add our own suite of testing verbs to a story beyond the "Test me with..." features already described.

Here are some types of testing verbs that can be useful to add:

Chapter jumps. We might create test commands that took us to a later stage of the story (perhaps doing more setup than "Test me..." alone can handle).

Status information. We might create a test command that would show us status information beyond what's covered in the Standard Rules. For instance, if we had a story that heavily modeled the moods of other characters and we wanted to be able to check those moods at any time, we might add a SHOWMOOD command that would tell us about a character's emotional state.

Puzzle satisfaction lists. Some simulation-rich stories offer puzzles that can be solved in a variety of ways: for instance, a sealed glass box that can be smashed with any object that has been marked with the properties "hard" and "heavy". Later, we might want to be able to check which in-story objects would work as a solution to this puzzle, so we might create a command like

```
Listing hammers is an action out of world applying to nothing.

Understand "list hammers" as listing hammers.

Carry out listing hammers:
    say "These things can break the glass: [line break]";
    repeat with item running through portable hard heavy things:
        say "[item][line break]";
```

so that we can review that there are enough objects available and that the list doesn't include anything it shouldn't. In a small story this kind of thing is pretty easy to keep track of in the author's head. Large stories can contain thousands of objects, however, at which point it becomes valuable to have an automated method of verification.

Just occasionally, we might also want to build a version of a story that will allow beta-testers access to the debugging commands. This is especially relevant for long stories: if we're testing a story with a lot of playtime and the testers have already thoroughly reviewed the first portion of the story, we might want to let them have access to testing commands that fast-forward to later sections.

To do this, we can use the "Release for Testing" feature. Release for testing builds a version of the story that *does* include testing commands and any sections labeled "Not for release".

Start of Chapter 24: Testing and Debugging
Back to §24.4. Low-level debugging commands

Onward to §24.6. Testing for thoroughness

§24.6. Testing for thoroughness

The presence of actual bugs or defects is not the only thing we want to consider when testing a story. We may also want to check whether we have built the story with a consistent amount of depth.

Are there descriptions for everything the player might look at? If we've implemented special verbs, do they have appropriate reactions for all the different objects? If most objects in a story about restaurant reviewing have a special response to being tasted, for instance, it might be disappointing for the player to encounter late-added objects that don't.

Checking implementation thoroughness can be a laborious process, but there are a few things we can do to automate it. For instance, we might add to a not-for-release section a rule that checks for certain properties:

```
When play begins (this is the run property checks at the start of play
rule):
    repeat with item running through things:
        if description of the item is "":
            say "[item] has no description."
```

This will confront us with a reminder of what we still need to fill in every time we start up the story.

There are also some extensions that are designed to assist with this, notably the massive Object Response Tests by Juhana Leinonen. Object Response Tests allows us to try out a long list of commands against any object in the story, so that we can quickly identify ones with nonsensical replies.

Start of Chapter 24: Testing and Debugging

Back to §24.5. Adding new testing verbs and Release for Testing

Onward to §24.7. Commands for beta-testers

§24.7. Commands for beta-testers

Inform includes a command that is especially designed to help beta-testers report flaws: namely, TRANSCRIPT. A tester can type TRANSCRIPT (or just SCRIPT) at the beginning of the story in order to start generating a recording of everything that happens. She can then add her own annotations when something buggy or otherwise notable occurs (for instance by typing a standard symbol, such as *, followed by a note).

When she then sends us the completed transcript, we can look through for these symbols and note the problems the tester found in the context of the rest of the story's behavior. Having information about how she reached that position typically makes it much easier to reproduce the problem than if she gave only a general account of it.

Start of Chapter 24: Testing and Debugging

Back to §24.6. Testing for thoroughness

Onward to §24.8. Help from the user community

§24.8. Help from the user community

Sometimes we get really stuck on a problem and despite all our best efforts cannot figure out how to solve it.

Fortunately, Inform has a lively and helpful community of users who are often willing to assist other authors. The easiest way to reach these users is to make a post at the intfiction forum at

http://www.intfiction.org/forum

and in particular to post Inform-related problems under the topic Inform 6 and 7 Development. Where possible, it's a good idea to post the example source that is causing trouble, and to make it as short as possible so that prospective helpers will not have to read any more than necessary in order to pinpoint the problem.

The user community is also a good place to find beta-testers who can try out our work and give feedback.

Start of Chapter 24: Testing and Debugging
Back to §24.7. Commands for

beta-testers

Onward to Chapter 25: Releasing:
§25.1. The finished product

Chapter 25: Releasing

Inform with Vorple; *§25.13. Website templates*; *§25.14. Advanced website templates*; *§25.15. Republis*

Contents of *Writing with Inform*
Chapter 24: Testing and Debugging
Chapter 26: Publishing
Indexes of the examples

§25.1. The finished product

This chapter and the next are about what to do when we have a complete, finished work on our hands.

For almost all of the time when a new work of IF is being written, it lives inside the familiar two-panel spread of the Inform user interface. But that isn't how eventual players will experience it. They will want to play a "story file" in a standard format, and they will do so with a wide range of different interpreters on many different computers or websites, including some -- like mobile phones -- on which Inform itself will not run.

So how does a new work of IF reach players? The simple answer, covered in this chapter, is that clicking the Release button instead of Go causes Inform to output a stand-alone story file. But as we will see, Release can do much more than that: it can attach covers, include bibliographic data, make websites and much more. Releasing is the process of making all of the material we want to deliver to our eventual players.

But that is only the first step. What do we do with the material when we have it? Printing out a manuscript of a novel is not the same as publishing it. So the next chapter, on Publishing, completes the story.

Start of Chapter 25: Releasing
Back to Chapter 24: Testing and Debugging: §24.8. Help from the user community
Onward to §25.2. Bibliographic data

---

§25.2. Bibliographic data

Almost all printed books have a title page and a so-called "imprint" page, often its verso, which make up a description of the contents. The title page gives the name of the book and of the author, while an imprint page contains a variety of details about the edition, the printing, and so on. An ISBN number is allocated so that, from the number alone, any book seller or cataloguer can identify exactly this work. Sometimes other cataloguing information is added, such as the Library of Congress classification. This set of information is called "bibliographic data", and without it libraries and booksellers would be at a total loss.

IF has bibliographic data, too. Inform has a number of special named values to hold this - who wrote the work being created, what it is called, what headline it has, what genre it has and what its release number is, and so on.

These can be set as follows:

```
The story title is "Mansfield Perk".
The story author is "Janet Austen".
The story headline is "An Interactive Romance".
The story genre is "Romance".
The release number is 7.
The story description is "In Miss Austen's new interactive novella,
Miss Henrietta Pollifax is adopted by the tempestuous landowner Sir
Tankerley Mordant, and must make a new life for herself on the
rugged moors."
The story creation year is 2005.
```

Most of these are self-explanatory. The "story creation year" is provided so that if we need to revise the work to fix some bugs a year later - by no means an uncommon occurrence - then we can make sure it is correctly identified as still being basically a 2005 work. (Just as a book which has had innumerable revised printings may say "First published 1988" on its imprint page.) The "story description" is a piece of text, analogous to the back cover blurb on a book: it might be two or three paragraphs long, so the example above is rather minimal, but it should not be epic in length.

As we have already seen, a convenient abbreviation provides that if the first sentence of the source text consists solely of text in quotation marks, then that is considered the title. Thus if the source begins:

```
"Mansfield Perk"
```

then that will be the "story title". Further, we can write

```
"Mansfield Perk" by Janet Austen
```

with the obvious effect: quotation marks around the author's name are optional here, for convenience, but note that we'd better have them in cases like:

```
"Three Men in a Boat" by "Jerome K. Jerome"
```

as otherwise the full stop after the K will end the sentence prematurely.

The text of these bibliographic descriptions cannot normally include text substitutions, since they are written into external descriptions of the story file as part of its "binding". Two exceptions are allowed, though: "[']" makes a literal apostrophe, and can be used if we need to override Inform's normal conventions to do with converting apostrophes at the ends of words to double-quotes. For instance:

```
"Summer of [']69" by Buzz Aldrin
```

The other exception is that the "[unicode ...]" text substitution works, so for example:

```
The story description is "This is a sentence[unicode 8212]with a
parenthetical in dashes[unicode 8212]because 8212 is the Unicode
number for an em-dash. But for example, 'pawn to [unicode black
chess bishop]4' draws in a black chess bishop, so it works with
names, too."
```

If the bibliographic named values are not set by the source text, Inform will still need to say something. Unset text and number variables evaluate to "" and 0 respectively, but this would make for a very unhelpful record. So Inform uses the following table instead of any value which is unset:

```
Story title: Untitled
Story author: Anonymous
Story headline: An Interactive Fiction
Story genre: Fiction
Release number: 1
```

Start of Chapter 25: Releasing

Back to §25.1. The finished product

Onward to §25.3. Genres

§25.3. Genres

The "story genre" is not used in the banner at all, and exists purely to help librarians. If it is at all possible to do so, authors are asked to use one of the following standard categories:

Comedy, Erotica, Fairy Tale, Fantasy, Fiction, Historical, Horror, Mystery, Non-Fiction, Other, Romance, Science Fiction, Surreal

These categories are based on those currently used by bookshops, but a few notes may be helpful. "Fiction" is intended for works whose essential purpose is literary, in a way which trumps any subject they happen to have: if Julian Barnes writes a mystery, for instance, a bookshop will shelve it with modern novels rather than in the detective stories section, whereas P. D. James's Adam Dalgliesh mysteries will end up filed with detective fiction even though she has appreciable claims to be an important novelist.

"Comedy" is used rather than "humour" to avoid the clash of spellings with "humor". This genre includes parodies.

"Non-Fiction" would be used for a work of IF which is essentially a presentation, perhaps in a novel interactive format, of true information. A meticulous simulation of the Great Exhibition of 1851, for instance, might qualify.

The distinction between "Surreal" and "Other" is that "Surreal" works contain at least some semblance of narrative, whereas "Other" is intended for works which "abuse" the format to present some entirely different sort of game - Tetris, say, or Minesweeper.

Start of Chapter 25: Releasing
Back to §25.2. Bibliographic data
Onward to §25.4. The Library Card

§25.4. The Library Card

Bibliographic data is useful for two reasons. Firstly, it enables the equivalent of a title page to be printed - traditionally called the "banner" - at the start of play; secondly,

Inform uses it to generate the equivalent of a library card for the work, which can be used by other programs to help organise, sort and classify interactive fiction. If the card is given to any other program on any other machine (or an Internet-based archive) then, in principle, that system can know about our work of fiction without a human librarian having to get hold of a copy, play it and laboriously copy out the details.

The "library card" is not of course a physical card, but a small "metadata" file which could potentially be transmitted quickly across the Internet. It contains no personal data other than what you choose to put on it, using the sentences documented in this chapter: it does not, for instance, identify your computer or IP address. In any case Inform does not send it anywhere, but merely keeps an up-to-date copy within the project, and includes it when making a release copy of the work. You can always see (a representation of) the current library card for a project in the Contents index.

Authors are asked to play fair, in return, by writing sensible and useful bibliographic information for any work which is likely to circulate to other people; by being honest (writing under a pseudonym is fine, but not impersonating other people); and by conforming to standard practice.

The Settings panel of each project contains a tick-box called "Bind up into a Blorb archive on release", and by default this is ticked. "Blorb" is a nonsense word from a popular early 1980s work of IF called "Enchanter", where it was the name of a magic spell whose purpose was to "safely protect a small object as though in a strong box". In the late 1990s, the name was borrowed for a standard format for what might be called the wrapping and packaging of IF. A typical Blorb archive produced by Inform contains the "story file" - the actual program for the story - together with its library card and cover art.

Modern IF interpreters such as Zoom for Mac OS X and Unix, and Windows Frotz, can play blorb archives directly, and the authors of Inform hope to make this the normal practice in future. Still, some interpreters cannot read blorbs directly and have to be given the actual story file: so by unchecking the above tick-box, we can insist that Inform creates only that. The disadvantage with this, of course, is that the library card (with all its bibliographic data) and any cover art is lost in the process.

Start of Chapter 25: Releasing

Back to §25.3. Genres

Onward to §25.5. The Treaty of Babel and the IFID

§25.5. The Treaty of Babel and the IFID

During March and April 2006, an agreement was reached between the IF archive and most of the different systems for creating IF - of which Inform is only one - called the

Treaty of Babel. While these different systems create computer programs which are quite different internally, the Treaty provides for works of IF to come with bibliographic data which identifies them in a standard way.

Inform is fully compliant with the Treaty. In particular, each new project created by Inform is allocated a unique identification number called its IFID. The IFID is the equivalent for IF of the ISBN of a printed book. Inform copies it onto the "library card" for the benefit of Internet-based libraries which may eventually accession the work. Of course many projects start but never see the light of day, so many possible IFIDs are "wasted": but that hardly matters, as there are plenty more numbers in the world.

The important thing is that

> **The IFID number must be unique to this one work out of all the IF ever created**

Inform will make sure this is true, **unless** we do something to break this ourselves. For instance, if we take an existing project, copy it as a file, then work divergently on the original and on the copy so that they become two radically different works, they will still each have the same ID. This is a bad thing: if we want to duplicate a project but then turn it into something new, the best way to do that is to create a new project, and to copy and paste the source from the old to the new.

Start of Chapter 25: Releasing

Back to §25.4. The Library Card

Onward to §25.6. The Release button and the Materials folder

§25.6. The Release button and the Materials folder

Inform's Release button does two things: it makes a stand-alone, public version of the current project - a "story file" - and it gathers up, or creates, whatever material we want to go with it.

The release version of the project can be played by anyone with an "interpreter" - they do not need the Inform application installed on their computers, and they will not be able to see the source text. Released versions differ slightly from the versions playable in the Story panel of Inform, because debugging commands such as ACTIONS are not included with them. (As we've seen, also excluded is any material in the source text under a heading including the words "not for release".) In some cases, if we release along with an interpreter, we can even make the project playable from a web browser, so that the player doesn't need to install any software at all, not even a browser plugin.

The Release button also creates a ".materials" folder for the project, if one doesn't exist already. (On some platforms, the Inform user interface creates it automatically alongside the project.) Inform adopts the following convention:

> The files associated with the project "Whatever.inform" should all be kept in a subfolder called "Whatever.materials" in the same folder that contains the project.

For example, if we have a project filenamed Magician.inform which lives in a folder called "Works in Progress", then files might be arranged like so:

```
Works in Progress
    Magician.inform
    Magician.materials
        Collegio.pdf
        Mating Wyverns.mp3
```

Of course "Magician" might not actually be the title of the project - it might be an abbreviation, or a working title. The name of the .materials folder has to match the name of the .inform file, not the title.

Several advanced features of Inform make use of the materials folder, and the "Release" button is one of them. It creates a further subfolder called "Release" within the materials folder. This is where it will always place the story file it creates, together with anything released "along with" the story - Inform will not need to put up a dialogue box asking us where to save the story file, because there is already a natural place. For instance, after a successful click on Release, we might then see:

```
Works in Progress
    Magician.inform
    Magician.materials
        Collegio.pdf
        Mating Wyverns.mp3
        Release
            Collegio.pdf
            Magician.zblorb
            Mating Wyverns.mp3
```

where "Magician.zblorb" is the actual story file produced by Inform. Note that Inform has made copies of the files to be released with it - the idea is that the Release subfolder contains only what Inform makes, and everything in the Release subfolder can be thrown away at any time.

This is especially useful if we're releasing along with a website (see below), as then the Release subfolder will be exactly what needs to be uploaded to a server to be shown to the world. Equally, the Release subfolder is what can be zipped up and uploaded to archives or (if small enough) emailed out.

Start of Chapter 25: Releasing

Back to §25.5. The Treaty of Babel and the IFID

Onward to §25.7. The Joy of Feelies

§25.7. The Joy of Feelies

"Feelie" is a slang word, again going back to the early days of IF, for something tactile included with commercially sold copies of IF games. For instance, Infocom's "Wishbringer" was not just a diskette in a pretty box: the box also contained a map, a letter, an envelope, a magic stone (well, a stone) and a booklet. Most of this was purely for fun, and to flesh out background to the story, but there would usually be clues sneaked into the text or artwork as well.

Today's IF is usually not supplied in physical packaging, and not accompanied by physical objects. But authors do sometimes want to include extraneous matter, whether it's a simple read-me file of instructions or a multimedia extravaganza. Inform does not provide facilities to make artwork, movies, soundscapes, booklets, etc.: there are plenty of programs out there to do all of that already.

But Inform does help with the collation and packaging-together. For instance, by placing the following sentence in the source text:

```
Release along with a file of "Collegio magazine" called "Collegio.pdf"
and a file of "The mating call of the green wyvern" called "Mating
Wyverns.mp3".
```

...we tell Inform that we will also be providing two additional files. Note that in each case we supply a brief description and a filename. The filename should always have a standard file extension for a well-known and thoroughly standardised file format - ".pdf" and ".mp3" are pretty safe: so for instance are ".txt", ".png", ".jpg", ".html". The filename should not include punctuation marks other than the full stop dividing name from extension, and should not exceed 30 characters in length.

It is also possible to supply a feelie which is not a single file, but is a mini-website: that is, a collection of interlinked HTML (and perhaps other) files. The convention here would be:

```
Release along with a file of "Baltrazar's Guide to Magic" called
"Guide".
```

The absence of a file extension on the filename "Guide" tells Inform that the feelie in question is a mini-website: it is expected to sit inside a folder called "Guide", with its home page being "Guide/index.html".

We have seen that Inform takes the story file, which is analogous to the pages of a book, and places it into a Blorb archive, analogous to the binding. These new additional files are not placed in the Blorb, because that would make the Blorb archive rather large (and would hide them from the player, which defeats the purpose). But references to them do appear in the Blorb, so that any interpreter playing the Blorb would be able to tell that there are supposed to be additional files available. Similarly, references are entered onto the library card.

Start of Chapter 25: Releasing

Back to §25.6. The Release button and the Materials folder

Onward to §25.8. Cover art

§25.8. Cover art

Accompanying files are not the only things which can be included in a "release along with" sentence: for instance, we could

```
Release along with cover art ("A stone gargoyle"), a file of "Collegio
magazine" called "Collegio.pdf" and a file of "The mating call of the
green wyvern" called "Mating Wyverns.mp3".
```

Cover art can not only be used to advertise a work of IF, it is also displayed to players by certain interpreters, such as Zoom or Spatterlight for OS X, or Windows Frotz for Windows. It is also used on the IFDB (ifdb.tads.org), and by browsing applications. If Zoom is installed, then on Mac OS X Leopard, the Finder shows cover art directly:

Cover art for a work should be prepared in either JPEG (".jpg") or PNG (".png") format, and we recommend that it should be square, like a music album cover. Programs which notice the cover art for a work of IF are likely to scale this up or down as convenient for their own display purposes, but it would be helpful to provide the original art at 960 by 960 resolution. (Remember, it is intended for use on screen, so providing art at magazine full-page print quality will add nothing and will only make the resulting release an annoyingly large download: even as it is, 960x960 will print out as a pretty respectable CD insert sleeve.) The cover art must not be smaller than 120 pixels in either dimension.

To provide cover art, we should create two files: Cover.jpg (or .png), and a reduction of the image to a smaller "thumbnail" version called Small Cover.jpg (or .png). These should be placed in the project's .materials folder. For instance, we might have:

```
Works in Progress
   Magician.inform
   Magician.materials
      Collegio.pdf
      Cover.jpg
      Mating Wyverns.mp3
      Small Cover.jpg
```

(supposing that, as in the previous examples, "Collegio.pdf" and "Mating Wyverns.mp3" are the filenames of two feelies that accompany the release). "Small Cover.jpg" should be a reduction of the main cover art image "Cover.jpg" to exactly 120 by 120 pixels.

The text in brackets after the release instruction...

```
Release along with cover art ("A cathedral at sunset.").
```

...is provided for the benefit of blind or partially sighted users, and should be brief.

§25.9. An introductory booklet and postcard

When IF is aimed particularly at people who have never played IF before, there are certain conventions which it's a good idea to explain, or players will simply not know what to do. It can become a chore writing a clear set of instructions, and then there is the further nuisance of explaining about the need for an interpreter program to play the IF story file.

To alleviate this, Inform can "Release along with an introductory booklet", as for instance in this example:

```
Release along with cover art, the introductory booklet, a file of
"Collegio magazine" called "Collegio.pdf" and a file of "The mating
call of the green wyvern" called "Mating Wyverns.mp3".
```

The introductory booklet is a standard 8-page PDF file, written and designed by Emily Short, which contains all the basic information needed for a player to get started. It has been written to be as general-purpose as possible, in the hope of being useful for a range of widely different works of IF. There will certainly be works to which it would not be an appropriate supplement, and some authors will certainly prefer to write their own notes for players, but of course it is not compulsory. By making it available as a convenience, the authors of Inform do not intend to say that these are the "official" instructions or that others are not. It is simply intended as a time-saver.

As an alternative, or a supplement, it's also possible to:

```
Release along with an introductory postcard.
```

which supplies a standard postcard about IF (everything new players need to know, at one glance) written and designed by Andrew Plotkin and Lea Albaugh.

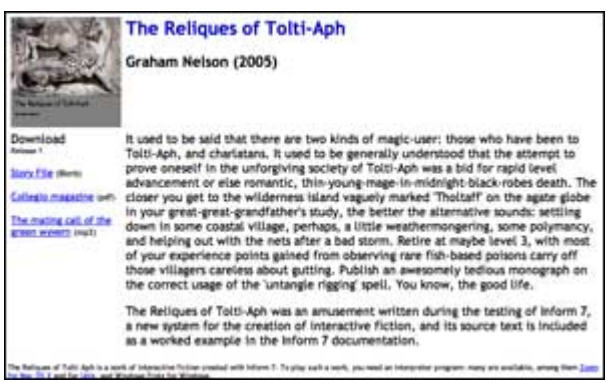
§25.10. A website

Much of the published IF of the last twenty years came with a brief text file describing what it was - a release note. Today it makes more sense to write this as a small web page, which can either be placed online, or simply distributed as part of the release.

Inform is able to manufacture such a website automatically. We request this by writing, for instance,

```
Release along with cover art, a website, a file of "Collegio magazine"
called "Collegio.pdf" and a file of "The mating call of the green
wyvern" called "Mating Wyverns.mp3".
```

where the list of ingredients now includes "a website". In fact, Inform makes only a single web page, called "index.html", which it places in the materials folder (as set up in the previous section): this then contains suitable links to all the other material, such as the cover art images, if they are also provided. For instance:



After a successful release now, then, we should see:

```
Works in Progress
    Magician.inform
    Magician.materials
        Collegio.pdf
        Cover.jpg
        Mating Wyverns.mp3
        Release
```

Collegio.pdf
Cover.jpg
index.html
Magician.zblorb
Mating Wyverns.mp3
Small Cover.jpg
Small Cover.jpg

Start of Chapter 25: Releasing
Back to §25.9. An introductory booklet and postcard
Onward to §25.11. A playable web page

§25.11. A playable web page

Modern web browsers are now so powerful as computing environments that they almost amount to general-purpose computers in their own right. The websites made in the previous section were passive, and simply displayed information about a story file. But it's also possible to make a more active page - one which can play the story file, right inside the browser, for anybody who visits.

To make such a page, we must:

Release along with an interpreter.

This automatically releases along with a website as well, since we need the website in order to house the new page, which will be called "play.html". This page will be bundled up with a customised copy of a story file interpreter coded in Javascript - in effect, a program for a web browser to follow - and a suitably encoded version of the story file. The practical effect should be that anyone visiting the page with any modern browser can just play.

Inform ships with the "Parchment" and "Quixe" interpreters built in, one intended for each of the story file formats it can output - so by default Inform uses Parchment if the format (on the project's Settings panel) is set to Z-code, and Quixe if the format is Glulx. But Inform also supports the use of any other interpreter the author wants to try (including, for instance, later versions of Parchment or Quixe than the built-in ones). If we have access to an exotic Javascript-based interpreter called, let's say, "Urbzig", then we can install it by putting it into the "Templates" subfolder of the ".materials" folder for the project, and then ask for it to be used instead of "Parchment" like so:

Release along with the "Urbzig" interpreter.

Start of Chapter 25: Releasing
Back to §25.10. A website
Onward to §25.12. Using Inform with Vorple

§25.12. Using Inform with Vorple

"Vorple" is an innovative system by Juhana Leinonen for allowing web-based Inform stories to make use of web controls and other gadgets. Using Vorple, a story can in principle have an entirely different user interface, and can make much better use of CSS styling, interface to Javascript libraries, and so on.

At time of writing, "Vorple" is limited to Z-machine stories only, but it's hoped to bring it to Glulx stories too by 2015. Further information can be found at its home page:

http://vorple-if.com

In the mean time, users can get started with Vorple by downloading the Vorple extensions (available on the Public Library), and by using the release instruction:

Release along with the "Vorple" interpreter.

which is included with the current build of Inform.

Start of Chapter 25: Releasing
Back to §25.11. A playable web page
Onward to §25.13. Website templates

§25.13. Website templates

Web pages are very idiosyncratic things and Inform will almost certainly not produce exactly what we want. What it actually does is to take an existing "template" web

page, and paste in the relevant information to make the final product. So by starting with a different template, we can end up with an entirely different-looking web page: like this one, for instance -

The template ordinarily used by Inform is called "Standard" and comes built in. (A second built-in template, "Classic", imitates the look used in 2005-08. The word "classic" here is to be understood in the sense of Classic Mac OS, the classic Doctor Who adventure "Time and the Rani", classic Mayan civilisation, and so forth - really pretty awful.)

Any other templates we must make ourselves, giving each one a different name, by convention a single word. In this section, we'll make a new one called "Platinum".

Suppose we write:

```
Release along with cover art, a "Platinum" website, a file of "Collegio
magazine" called "Collegio.pdf" and a file of "The mating call of the
green wyvern" called "Mating Wyverns.mp3".
```

This is identical to the previous version except for the "Platinum": note the quotation marks. When it needs to find a template, Inform searches the following places in sequence:

```
(a) the "Templates" subfolder of the project's own .materials folder, if
this subfolder should exist;
(b) the "Templates" folder in the user's own library - on Mac OS X,
this is:
    ~/Library/Inform/Templates
or on Windows:
    My Documents\Inform\Templates
or on Linux:
```

```
    /Inform/Templates
(c) the built-in stock of templates, currently only "Standard" and
"Classic".
```

What Inform looks for is a folder name matching that of the template - so in our case we need to provide a folder called "Platinum", and put it in either location (a) or (b).

The template folder is expected to contain some combination of the following files:

```
Platinum
    index.html
    source.html
    style.css
    (extras).txt
```

There are two HTML pages here, one for the main front page, the other for pages of displayed source text (if we release along with the source text - see later in the chapter). The CSS file defines styles of text - sizes, fonts, colours, and so on - and positions material on the page. The "(extras).txt" - which is optional, of course - allows additional HTML pages, images, movies and so on to be added.

If any of these is missing, Inform uses the one in "Standard" instead. In practice, this means the easiest way to create a new template is to supply just a new CSS file, which can change the colour, font, type size, and position of more or less everything in the site:

```
Platinum
    style.css
```

We probably want to start from the "Standard" version of "style.css" and edit in a few changes; the easiest way to get a clean copy of "Standard"'s CSS file to work on is to release the project with a "Standard" template, which causes this default "style.css" to appear in the "Release" subfolder of the project's .materials folder. (But it's wise to move the file out of "Release" before starting to edit it - files in "Release" are overwritten by Inform whenever a release is made.)

This is not the place to describe how CSS works. CSS is a more or less universal format today for describing how web pages should look - their style rather than their content. A dazzling variety of possibilities can be seen at the excellent:

```
www.csszengarden.com
```

but of course there are many, many other textbooks and websites which describe

CSS.

Start of Chapter 25: Releasing
Back to §25.12. Using Inform with Vorple
Onward to §25.14. Advanced website templates

§25.14. Advanced website templates

The following describes how Inform uses the extras file and the two HTML pages in a template, and will only be needed if a new template has to make changes so radical that altering the CSS alone won't be enough.

The optional "(extras).txt" file - note brackets - is a text file which contains a list of named extras to throw in. For instance:

```
easter.html
egg.png
```

These named files need to be present in the template folder. Files with the extension ".html" go through the placeholder expansion process just like the index and source pages; all other files are copied verbatim.

HTML templates like "index.html" and "source.html" are fully valid HTML pages in their own right, though they have placeholder text where Inform will substitute the project's bibliographic data (see below). The "<head>" element should include a reference to "style.css", which of course will mean the CSS file given in the template (or the one from "Standard" if no CSS file is given) - for instance,

```
<link rel="stylesheet" href="style.css" type="text/css" media="all" />
```

When it turns the template into the final web page, what Inform does is to replace certain capitalised words in square brackets with the appropriate text:

```
[TITLE] becomes the story title
[AUTHOR] becomes the author's name
[YEAR] becomes the story creation year
[BLURB] becomes the story description
```

[RELEASE] becomes the release number
[COVER] becomes an image of the cover art (the small 120x120 cover image)
[DOWNLOAD] becomes the download link
[AUXILIARY] becomes the list of feelie-like files, if any
[IFID] becomes the IFID
[STORYFILE] becomes the "leafname" of the story file, e.g., "Bronze.gblorb"
[TEMPLATE] becomes the name of the template used to make the page
[SMALLCOVER] becomes the filename of the small cover image
[BIGCOVER] becomes the filename of the large cover image
[TIMESTAMP] and [DATESTAMP] become the time and date of releasing

Everything else is left alone. In source pages, five further placeholders are available:

[SOURCE] becomes the portion of the source text on this page
[SOURCELINKS] becomes the navigational links
[SOURCENOTES] becomes the footnote matter at the bottom of the source
[PAGENUMBER] and [PAGEEXTENT] are such that the text "page [PAGENUMBER] of [PAGEEXTENT]" produces, e.g., "page 2 of 7"

Both [SOURCE] and [SOURCENOTES] must exist on the page, and [SOURCENOTES] must appear after [SOURCE] does in the file. (Of course the CSS in "style.css" might move the copy around on screen, but that's another matter.)

Start of Chapter 25: Releasing
Back to §25.13. Website templates
Onward to §25.15. Republishing existing works of IF

§25.15. Republishing existing works of IF

Some long-time users of Inform will have projects which were originally made using the very different Inform 6 language. Story files produced with Inform 6 do not have any of the extra touches in this chapter: in particular, they have no cover art and no bibliographic data, which makes them rather plain and anonymous to newer Treaty of Babel-equipped programs like Zoom, Spatterlight or Windows Frotz.

To help with this, today's Inform can republish an Inform 6 project by combining an Inform 7 source text which contains only release instructions and bibliographic data

with an already-compiled Inform 6 story file. We do this by writing a short source text which contains:

```
Release along with an existing story file.
```

We then place the story file in the ".materials" folder. By default this will be called "Story.z8", but we can alternatively name it:

```
Release along with an existing story file called "Zork1_sg.z5".
```

The Settings panel must be switched to the Z-machine for this to work, since only Z-machine story files are supported this way, not Glulx. And we can now use the Release button to obtain the goods.

An existing story file can take advantage of all of the extra features - cover art, titling, website, feelies and so forth - earlier in this chapter, but not those - walkthrough, source text, map - which are still to come.

The following is a typical example of a source text used solely to bind up an old Inform 6-compiled story file:

```
"Curses" by Graham Nelson

The story genre is "Fantasy".

The story headline is "An Interactive Diversion".

The story creation year is 1993.

The release number is 16.

The story description is "It's become a matter of pride now not to give
```

```
up. That tourist map of Paris must be up here somewhere in all this
clutter, even if it has been five years since your last trip. And it's your
own fault. It looks as if your great-grandfather was the last person to
tidy up these lofts..."


Release along with cover art and an existing story file.
```

Start of Chapter 25: Releasing

Back to §25.14. Advanced website templates

Onward to §25.16. Walkthrough solutions

§25.16. Walkthrough solutions

Since the earliest days of IF, players have distributed solutions to well-known stories, to help out other players at their wits' ends. The commonest format for these is a list of commands to type, sometimes with notes in the margin, and such a solution is called a "walkthrough", since it walks a player through the story.

Few authors publish solutions of their own works, but many supply their testers with solutions, especially towards the end of testing, or submit a solution as part of a competition entry. To help with this, Inform can generate such a walkthrough solution automatically:

```
Release along with a solution.
```

Inform will then place a file called "solution.txt" inside the "Release" folder. The solution might look like so (although probably much longer):

```
Solution to "Memoirs of India" by Graham Nelson


Choice:
INVENTORY -> go to branch (1)
EAST -> go to branch (2)
```

Branch (1)
DROP MANUSCRIPT
SOUTH

Branch (2)
INVENTORY ... Always a good idea
GIVE MANUSCRIPT TO THOMAS

Inform does not, of course, know how to solve IF all by itself, but derives the solution from the project's Skein. Since the Skein will have been used in testing the story, it will very likely contain a perfect solution - or several different ones, taking the story to a variety of possible endings. In the example above, there are two possible winning lines, which diverge right from the first move. (There can be further divergences: for instance, if branch (2) splits, it will split into branches called (2.1), (2.2), (2.3) and so on.)

But the Skein will also contain plenty of unwanted diversions, so Inform does not rewrite the entire Skein as a solution. Instead, it looks for knots in the Skein which have been annotated. Any knot whose annotation begins "***" (three asterisks) is considered to be a final, winning move. (It is probably a good idea to lock such a knot once it has been annotated thus, too.) We can mark any number of knots "***" since, after all, we can declare any number of lines of play as possible solutions. Inform then constructs the solution out of all lines of play in the Skein which lead to "***" endings, and ignores other threads.

Annotations other than "***" in the Skein are turned automatically into comments in the solution text. For instance, the knot for the INVENTORY command in the second branch above was annotated "Always a good idea", and this was transcribed into the solution. (If an ending knot is annotated with, say, "*** Happy ending!" then the "***" marks it as an ending, and "Happy ending!" is added as an annotation to that ending.)

By default, the solution text is not linked from our webpage, on the assumption that we may want to generate a walkthrough but not immediately advertise it to players. If

we wish to change this, we may write instead

```
Release along with a public solution.
```

The terms public and private may also be applied to other elements we are having Inform generate to include on our webpage: see also the notes on private source text, below.

Start of Chapter 25: Releasing

Back to §25.15. Republishing existing works of IF

Onward to §25.17. Releasing the source text

§25.17. Releasing the source text

Most authors will not want to publish the source text alongside the work itself, because this gives away all of its secrets. Inform provides the option mainly for the sake of the examples published on its own website, where making the source available is the whole point. But anyone is welcome to use the option, of course:

```
Release along with the source text.
```

If Inform is not also generating a website, this produces a plain text file called "source.txt" in the "Release" folder, and there is nothing more to be said.

However, if a website is also being released, the source is also converted to a suite of web pages which are linked to and from the home page. (Each heading with substantive content is placed on its own web page, with the opening page containing a contents list.)

Comments in the source are rendered in grey. As a special feature, any comment which begins with an asterisk is considered a footnote and is printed below the source text, with a link. Thus comments thus:

```
Hercules is a demigod.[* We're using Greek spellings so he ought to
be Heracles, but players are so much more familiar with Hercules.]
```

will be printed more like so:

```
Hercules is a demigod.[1]
...
Note
[1]. We're using Greek spellings so he ought to be Heracles, but
players are so much more familiar with Hercules.
```

Footnotes are automatically numbered from 1 on each source page.

By default, the source text is linked from our generated webpage, if we are releasing with a webpage. If we wish to change this, we may write instead

```
Release along with the private source text.
```

This will create a text file containing the source for our story, and place this file in our release folder, but not create a link so that the player can find it.

Finally, we can:

```
Release along with the library card.
```

which releases a stand-alone XML file in 'iFiction' format for the bibliographic data on the story file; this is the same data embedded in the blorb file itself, but having an external copy makes it easier to see what Inform has done, and some external programs can read iFiction data like this.

Start of Chapter 25: Releasing

Back to §25.16. Walkthrough solutions

Onward to §25.18. Improving the index map

§25.18. Improving the index map

As we have seen, "Release along with..." allows us to package up a work of IF with all manner of extra materials. But what are these to be? One popular option is to produce a map - sometimes partial, sometimes obfuscated - and supply that with the story: besides, there are some IF competitions where the rules require that the referee is supplied with a map even if the players are not, and failing that, it is sometimes nice to be able to print out a map of a work in progress.

The World map in the Index tab is heavily stylised and cartoonish, intended to be clicked on or moused over, and viewed in a browser: although it is, in fact, possible to print it, the results are not very good. Fortunately, the same underlying map mechanism can be used to output something more useful and very much more customisable, as we shall see.

The map-maker is one of the most complex parts of Inform, even though it actually contributes nothing to the final story file: the problem of how to draw up a "correct" map from the source text is by no means easy to solve. Inform tries, but it often gets things wrong. Its general practice is to place rooms on a square grid (actually a cubic lattice, as it works in three dimensions), but not all conceptual maps fit well onto this, and Inform often annoyingly puts a particular room in the "wrong" place. For instance, suppose Inform puts "Didcot" east of "Abingdon" and this makes the geometry look different to what we had in mind. We can correct with:

```
Index map with Didcot mapped southeast of Abingdon.
```

Note that this says nothing about exits from any room to any other room, and changes the final work of IF not at all: it simply helps Inform to draw the map index. (Instructions like this one are treated as being almost certainly true, but Inform does not quite always obey: it will never allow two rooms to be superimposed at the same grid position, no matter what we have asked in "Index map with..." instructions.) The same trick is useful if we have a situation like so:

```
Inside of Sweeping Sands is Beach Hut Interior.
```

"Beach Hut Interior" is a single room which does not connect to the rest of the map by any of the ten spatial directions, so Inform does not place it on the main map but instead moves it off out of the way in a map of its own. Given that it's just a single room, however, we might prefer to put into a convenient otherwise empty grid position like so:

```
Index map with Beach Hut Interior mapped west of Sweeping Sands.
```

Finally, note that this trick also ensures that the two locations are mapped on the same level vertically, and can be useful in cases where room A is both north of and above room B: Inform will want A to be higher up than B, but we can insist otherwise.

Start of Chapter 25: Releasing

Back to §25.17. Releasing the source text

§25.19. Producing an EPS format map

The "Index map with..." instruction is a much more varied thing than hinted at in the previous section, and its general form is

```
Index map with [instruction] and [instruction] and ... and [instruction].
```

where the instructions can be of four different forms, as follows:

```
[room A] mapped [direction] of [room B]
EPS file
rubric [text] ... and some optional details ...
[setting] of [whatever] set to [value]
```

We have already seen the first of these instructions. The second is short and has a fixed wording:

```
EPS file
```

so can be invoked by typing "Index map with EPS file.", for instance. EPS stands for Encapsulated PostScript, which is a standard file format for line art. EPS files can be edited with sophisticated graphics programs such as Adobe Illustrator, and can be used as illustrations in many word-processors and page layout programs. They can also be converted to PDF by Mac OS X Preview, or used in Linux or Windows with the open-source Evince viewer. We need a line-art format because the map produced will never be exactly what we want: we are probably going to end up hacking it to change the fonts, add some drawings, tidy up the spacing and so on. A really large map will end up using quite a large "canvas", in EPS terms; it may be necessary to shrink it down in order to get it onto an A4 page, or to adjust whatever editing software is used to "custom paper size".

When the map-maker has been given the "EPS file" instruction, it writes an attempt to draw the current project's map in EPS format as a file in the project's ".materials" folder, with the filename "Inform Map.eps".

Note that Inform will over-write any existing file of this name: but that is intentional, because one usually ends up tweaking and rebuilding the project over and over to get

the map just so, and it would be tiresome for Inform to produce endless copies "Inform Map 19.eps", etc.

(The reason the EPS file is not placed in the Release subfolder is that it is not going to releasable to the public as it stands: for one thing it will be too raw, and for another, EPS is not a format everyone can read. It is provided as raw materials.)

Start of Chapter 25: Releasing

Back to §25.18. Improving the index map

Onward to §25.20. Settings in the map-maker

§25.20. Settings in the map-maker

The map-maker has altogether 35 named settings, and tweaking these can affect the result in ways which vary from the subtle to the grotesque. An important point is that the map-maker deals separately with the three levels in its working: the big picture of the whole map; each of the vertical slices which contain sub-maps; and finally all of the individual rooms. For instance, we might have 67 rooms, arranged on 3 vertical levels, all shown on one big map: Inform will try to show these stacked above each other, with the highest level at the top of the map, then the middle level, then the bottom level.

Moreover, not only does the whole map have its 35 settings, but each level has its own independent collection of those 35 settings, and so does each individual room. So the actual number of variables in our example is 1+3+67 = 71 times 35, which is a lot. The convention is that setting the value of S (some setting, let's say) for something affects not only that thing, but also everything inside it, unless they have their own individual settings for S.

For example: one of the settings is called "room-size", and is the size of the little square boxes representing a room, measured in points. (One point is 1/72 of an inch, so 72 points equals 1 inch: it's a traditional printer's measure.) Suppose we write:

```
Index map with room-size set to 36
    and room-size of level 2 set to 28
    and room-size of the Hall of Kings set to 52.
```

The first instruction sets the value of "room-size" for the whole map (note the lack of an "of..."); the second for level 2 of the map, and the last for a single room only. The result is that the Hall of Kings is drawn as 52x52 point box, all rooms on level 2 are 28x28 (except the Hall of Kings, if it's on level 2), and all others are 36x36, half an

inch square.

The setting instruction also allows three other useful forms. A setting "of the first room" applies to the room in which the story begins: we might for instance write

```
Index map with room-outline-thickness of the first room set to 2.
```

which gives this special room a bolder edge to it, since the default value is 1.

We can also apply settings not just to single rooms but to all rooms of a given kind:

```
A rivery room is a kind of room. Index map with room-colour of rivery
rooms set to "Navy" and room-name-colour of rivery rooms set to
"White".
```

Lastly, we can apply settings to all rooms in a given region:

```
Northern Oxfordshire is a region. Hampton Poyle and Steeple Barton
are in Northern Oxfordshire. Index map with room-name-font of
Northern Oxfordshire set to "Helvetica-Oblique".
```

(Note that rooms and regions don't have their own individual sets of the 35 settings: what happens is just that instructions like the last one change more than one room at once.)

Start of Chapter 25: Releasing

Back to §25.19. Producing an EPS format map

Onward to §25.21. Table of map-maker settings

§25.21. Table of map-maker settings

Note that all map-maker settings have single word names, though many are hyphenated, and that "colour" is always given the English and Canadian spelling, not the American form "color".

font (named in double-quotes)
integer (measured in points: 72 = 1 inch)
text (in double-quotes)
integer (measured in points)
font (named in double-quotes)
colour (named in double-quotes)
on/off
integer (measured in points)
integer (spacing measured in points)
on/off
integer (measured in points)
integer (length to abbreviate down to)
font (named in double-quotes)
text (in double-quotes)
integer (measured in points)
font (named in double-quotes)
colour (named in double-quotes)
integer (measured in points)
integer (Bezier spline curve scale factor)
integer (measured in points)
colour (named in double-quotes)
offset (in percentages of grid-size)
integer (measured in points)
colour (named in double-quotes)
text (in double-quotes)
integer (measured in points)
font (named in double-quotes)
colour (named in double-quotes)
integer (length to abbreviate down to)
offset (in percentages of grid-size)
on/off
colour (named in double-quotes)
integer (thickness in points)
shape (named in double-quotes)

Start of Chapter 25: Releasing

Back to §25.20. Settings in the map-maker

Onward to §25.22. Kinds of value accepted by the map-maker

---

§25.22. Kinds of value accepted by the map-maker

Integer values are typed in the usual way: 3, -72, etc.

Text is in double-quotes: "Map of Lower Delta", etc.

Font names are in double-quotes: "Helvetica", etc. Note that Inform makes no effort to look for such fonts: if we give the name of a font we haven't got, the result will probably be that the map's EPS file will be displayed in various applications with Courier (which looks like bad typewriting) substituted. All fonts are by default equal to the global "font" setting (by default equal to "Helvetica"), so changing "font" for the whole map affects everything not explicitly specified as having a different font.

Shape names are in double-quotes with lower case. At present, the only legal shapes are "circle", "square" and "rectangle".

On/off values are written just thus: on, off. No quotation marks.

Offset values are actually pairs, and are written as two numbers (possibly negative numbers) joined by an ampersand, as in the example: "Index map with room-offset of Botley set to 10&-30." Note lack of spaces around the ampersand. This means that Botley's room is displaced from its correct grid position on the EPS map by 10% of the grid size eastwards, and 30% southwards. (The grid size is the distance between one grid position and the next: displacing Botley by -200&0 would move it two whole grid positions westwards.)

The **route-stiffness** setting is used when drawing routes between two rooms. These are drawn as Bezier curves, a standard way to make a smooth curve not only travel from A to B but also from pointing in a given direction at A to ending up pointing in a given direction at B. Thus a Bezier curve may turn a route round so that it leaves A pointing west, but curves around to enter B from the south. (Most routes involve leaving in one direction and arriving in the opposite direction, of course, and in those cases a Bezier curve is just a straight line.)

The stiffness factor for a given room measures how much the curves are allowed to warp around in order to force them to arrive at that room from exactly the right compass bearing. The default is 100. Raising to, say, 250 can force curved paths into freakish zig-zags: whereas lowering to 1, the minimum, may make the route arrive at completely the wrong bearing. (Formally speaking: at each end of the route, a "control point" for the Bezier curve is made by taking the centre point of the room, then adding the relevant compass bearing's vector, scaled up by the route-stiffness as a percentage of the grid size.)

Colour values are named and in double-quotes. These names are the same as those for the traditional set of web-page-safe colour chips, as follows:

"Alice Blue"
"Antique White"
"Aqua"
"Aquamarine"
"Azure"
"Beige"

"Bisque"
"Black"
"Blanched Almond"
"Blue"
"Blue Violet"
"Brown"
"Burly Wood"
"Cadet Blue"
"Chartreuse"
"Chocolate"
"Coral"
"Cornflower Blue"
"Cornsilk"
"Crimson"
"Cyan"
"Dark Blue"
"Dark Cyan"
"Dark Golden Rod"
"Dark Gray"
"Dark Green"
"Dark Khaki"
"Dark Magenta"
"Dark Olive Green"
"Dark Orange"
"Dark Orchid"
"Dark Red"
"Dark Salmon"
"Dark Sea Green"
"Dark Slate Blue"
"Dark Slate Gray"
"Dark Turquoise"
"Dark Violet"
"Deep Pink"
"Deep Sky Blue"
"Dim Gray"
"Dodger Blue"
"Feldspar"
"Fire Brick"
"Floral White"
"Forest Green"
"Fuchsia"
"Gainsboro"
"Ghost White"
"Gold"
"Golden Rod"
"Gray"
"Green"
"Green Yellow"
"Honey Dew"
"Hot Pink"
"Indian Red"

"Indigo"
"Ivory"
"Khaki"
"Lavender"
"Lavender Blush"
"Lawn Green"
"Lemon Chiffon"
"Light Blue"
"Light Coral"
"Light Cyan"
"Light Golden Rod Yellow"
"Light Grey"
"Light Green"
"Light Pink"
"Light Salmon"
"Light Sea Green"
"Light Sky Blue"
"Light Slate Blue"
"Light Slate Gray"
"Light Steel Blue"
"Light Yellow"
"Lime"
"Lime Green"
"Linen"
"Magenta"
"Maroon"
"Medium Aquamarine"
"Medium Blue"
"Medium Orchid"
"Medium Purple"
"Medium Sea Green"
"Medium Slate Blue"
"Medium Spring Green"
"Medium Turquoise"
"Medium Violet Red"
"Midnight Blue"
"Mint Cream"
"Misty Rose"
"Moccasin"
"Navajo White"
"Navy"
"Old Lace"
"Olive"
"Olive Drab"
"Orange"
"Orange Red"
"Orchid"
"Pale Golden Rod"
"Pale Green"
"Pale Turquoise"
"Pale Violet Red"

"Papaya Whip"
"Peach Puff"
"Peru"
"Pink"
"Plum"
"Powder Blue"
"Purple"
"Red"
"Rosy Brown"
"Royal Blue"
"Saddle Brown"
"Salmon"
"Sandy Brown"
"Sea Green"
"Sea Shell"
"Sienna"
"Silver"
"Sky Blue"
"Slate Blue"
"Slate Gray"
"Snow"
"Spring Green"
"Steel Blue"
"Tan"
"Teal"
"Thistle"
"Tomato"
"Turquoise"
"Violet"
"Violet Red"
"Wheat"
"White"
"White Smoke"
"Yellow"
"Yellow Green"

Start of Chapter 25: Releasing

Back to §25.21. Table of map-maker settings

Onward to §25.23. Titling and abbreviation

§25.23. Titling and abbreviation

The main title of the map is the value of "title" for the whole map, so for instance we might write:

Index map with title set to "Oxford and its Environs".

The subtitle settings apply to the subtitles used for each of the levels, so for instance

Index map with subtitle of level -1 set to "Tunnels and Sewers".

Names of individual rooms can be controlled with:

Index map with name of Radcliffe Camera set to "Library".

(By default, the name of a room is its name in the main IF project, of course.) The smallest writing on the map is normally that used to label unorthodox or unclear exits (in particular, those going from one layer to another): this is what the "annotation" size, font and colour are used for.

For most ways to set up the map, it's a practical necessity to abbreviate names of rooms, or they will spill out all over each other. Inform does this using the "room-name-length" setting. (The "annotation-name-length" is analogous.) For instance, if this setting is 5, then Inform will reduce the text of a name to at most 5 characters. It does this by successively throwing out spaces, lower case vowels, then other lower case letters, punctuation marks and finally upper case letters, always starting at the back of the name and working inwards: the process stops as soon as the name is short enough. For instance, "Reading" is abbreviated to "Redng", "Shangri-La" to "Shn-La" and "Cloud-Cuckoo-Land" to "C-C-L". The result can be a little comical, but is surprisingly unambiguous in practice. Abbreviation can effectively be abolished by raising the "room-name-length" to 128 (the highest permitted level), and note that the setting can be changed for individual rooms, so it is possible to have some room names abbreviated and others not, or in different degrees.

Start of Chapter 25: Releasing

Back to §25.22. Kinds of value accepted by the map-maker

Onward to §25.24. Rubrics

Example 446: ★ **Baedeker** Creating a floorplan of the cathedral using the locations from previous examples.

Example 447: ★ **Port Royal 5**

Port Royal scenario given instructions for an EPS map.

Example 448: ★ **Bay Leaves and Honey Wine** Creating a map of Greece using the locations from previous examples.

---

§25.24. Rubrics

Lastly, we can add our own arbitrary text to the map: perhaps to annotate points, perhaps just to add more heading matter (such as the author's name, or the date). Each individual line added - and only single lines can be added, not typeset paragraphs - is called a "rubric". (There can be up to 100 of these.) We can create a rubric like so:

```
Index map with rubric "Here Be Wyverns" size 16 font "Helvetica-
Oblique" colour "Thistle" at 150&0 from Cloud-Cuckoo-Land.
```

This gives rather more detailed information than is needed: "size 16" could have been omitted, giving us 12-point type by default, and similarly there is no need to specify a font unless it differs from the main "font" setting for the whole map; and the colour will be black if unspecified. The "at" position does need to be given, though. Note that it is relative to a given room on the map, and that the position specified is that of the centre-point of the text. (If we had written just "at 100&100", say, that would specify a position relative to the bottom left hand corner of the map.) So, for instance:

```
Index map with rubric "trapped door" size 8 at -60&-60 from Longwall.
```

would add a little 8-point-type safety tip for naive map-followers.

Inevitably, the settings in the map-maker will fail to get exactly the effect desired (though they will offer an excellent opportunity to waste entire days). But that is the whole point of producing output in EPS format: Inform aims not to produce final print-ready professional art, but to produce the raw material for making that final work of art. And if all that's required is a sketch-map, then Inform's output should be good enough quickly and without too much fuss.

Start of Chapter 25: Releasing

Back to §25.23. Titling and

abbreviation
Onward to Chapter 26: Publishing: §26.1. Finding a readership

Examples from Chapter 25: Releasing

Start of this chapter
Chapter 26: Publishing
Indexes of the examples

446 **ExampleBaedeker** *RB*

Creating a floorplan of the cathedral using the locations from previous examples.

If our map is largely or entirely set inside a single building, we might want to produce something that resembles a floorplan. It's possible to do this with a little tweaking, like so:

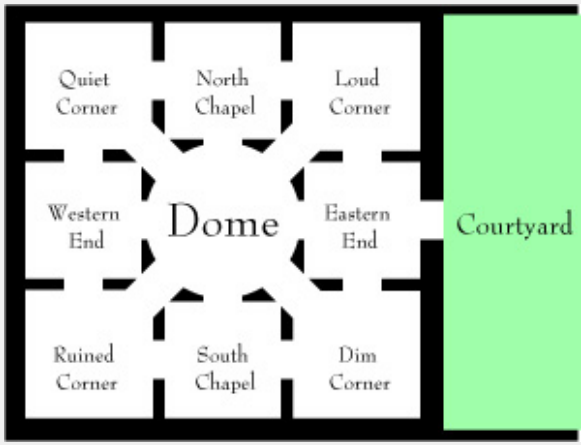
"Baedeker"

Dome is a room. North of Dome is North Chapel. South of the Dome is South Chapel. West of the Dome is Western End. Quiet Corner is northwest of the Dome, north of Western End, and west of North Chapel. Loud Corner is east of North Chapel, northeast of Dome, and north of Eastern End. Eastern End is north of Dim Corner and east of Dome. Dim Corner is southeast of Dome and east of South Chapel. Ruined Corner is southwest of Dome, west of South Chapel, and south of Western End.

The church door is east of Eastern End and west of the Courtyard. The church door is a door.

Index map with
room-shape set to "square" and
room-size set to 60 and
room-name-size set to 9 and
room-name-length set to 13 and
route-thickness set to 20 and
room-outline set to off and
map-outline set to off and
route-colour set to "White" and
room-colour set to "White" and
room-shape of Dome set to "circle" and
room-size of Dome set to 80 and
EPS file.

Now we have a map made of white lines and boxes over a white background, which is not very exciting. If, however, we put a layer of black under this and slightly adjust the room shapes (using an image editor such as Adobe Illustrator), we can produce something that plausibly resembles a floorplan or museum map, like so:



---

447 **ExamplePort Royal 5** *RB*
Port Royal scenario given instructions for an EPS map.

"1691"

Fort James is a room.

Thames Street End is south of Fort James.

Lime Street is south of Thames Street End. West of Thames Street End is north of Fisher's Row. The description of Fisher's Row is "A waterfront street that runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west Thames Street."

Water Lane is east of Thames Street End.

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Fishmarket is southwest of Thames Street at the Wherry Bridge.

The Private Alley is south of Thames Street at the Wherry Bridge.

Thames Street by the King's House is east of Thames Street at the Wherry Bridge.

Thames Street before Fort Carlisle is east of Thames Street by the King's House.

South of Thames Street before Fort Carlisle is a room called Fort Carlisle. The description of Fort Carlisle is "Handsomely arrayed with cannons which you could fire at any moment -- though of course there are ships at dock which might be in the way."

Queen Street End is south of Lime Street.

Queen Street Middle is east of Queen Street End.

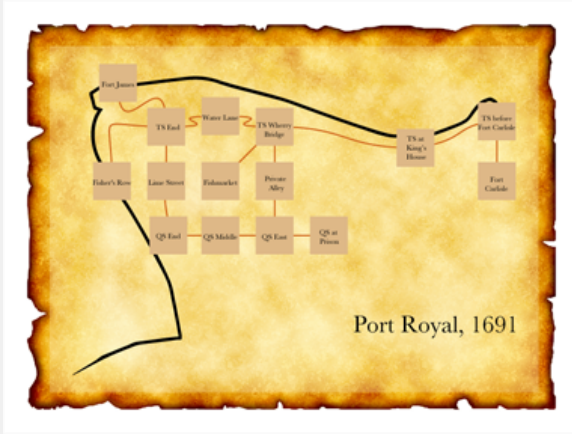
Queen Street East is east of Queen Street Middle and south of Private Alley.

Queen Street at the Prison is east of Queen Street East.

Index map with an EPS file and
    Fisher's Row mapped southwest of Thames Street End,
    room-size set to 50 and room-name-size set to 10,
    room-name-length set to 15,
    route-thickness set to 2,
    room-outline set to off,
    map-outline set to off,
    route-colour set to "Chocolate",
    room-colour set to "Burly Wood",
    title set to "Port Royal, 1691",
    font set to "Baskerville",
    room-offset of Thames Street by the King's House set to 160&-40,
    room-offset of Thames Street before Fort Carlisle set to 210&10,
    room-offset of Fort Carlisle set to 210&0,
    room-offset of Fort James set to -90&-20,
    room-offset of Water Lane set to 0&20,
    room-offset of Queen Street End set to 5&0.

This sentence has become a long catalogue of specifications. To break it down: we disambiguate the placement of Thames Street End, which otherwise is hard to locate because the directions to and from the room are not symmetrical. Then we apply some general rules about size, font, and color. Finally, we add instructions about offsetting the room locations of a few specific rooms.

This last part is a bit finicky and will not be necessary in many cases, but our goal this time is to create a map diagram that can be superimposed on the real coastal outline of Port Royal at the time. With a bit of editing, the result looks like this:



---

448 **ExampleBay Leaves and Honey Wine** *RB*

Creating a map of Greece using the locations from previous examples.

The map-maker can be used in quite versatile ways, in short; though the default is a schematic line-and-box affair, that is hardly the only option. While the EPS created is not always the result of our dreams, Inform usually can be made to do most of the hard and boring part, leaving the author to do only a bit of aesthetic touchup.

In many previous examples, we have sent hapless deities wandering around a map of Greece; we might like to chart that for ourselves, in a semi-realistic fashion. So:

"Bay Leaves and Honey Wine"

```
Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.



Index map with an EPS file and
    room-size set to 8,
    map-outline set to off,
    room-name-offset set to 40&-40,
    room-outline set to off,
    room-colour set to "White",
    route-colour set to "White",
    room-name-colour set to "White",
    room-name-length set to 25,
    room-shape set to "circle".
```

This produces a line-and-dot map, where the names of rooms do not appear inside the city-circles, but rather (thanks to "room-name-offset") off to one side. We specify a long room-name-length because we want all the names of the cities spelled out in full; and we make all the elements white because we intend to place them over a black background layer.

We can then superimpose this on a vector map of Greece and tweak the exact positions of cities a little by hand (in Adobe Illustrator, as it happens, but other programs would also allow this level of editing). The result:

DELPHI
THEBES
ATHENS
CORINTH
OLYMPIA
MYCENAE
ARGOS
EPIDAURUS
PYLOS
SPARTA

Chapter 26: Publishing

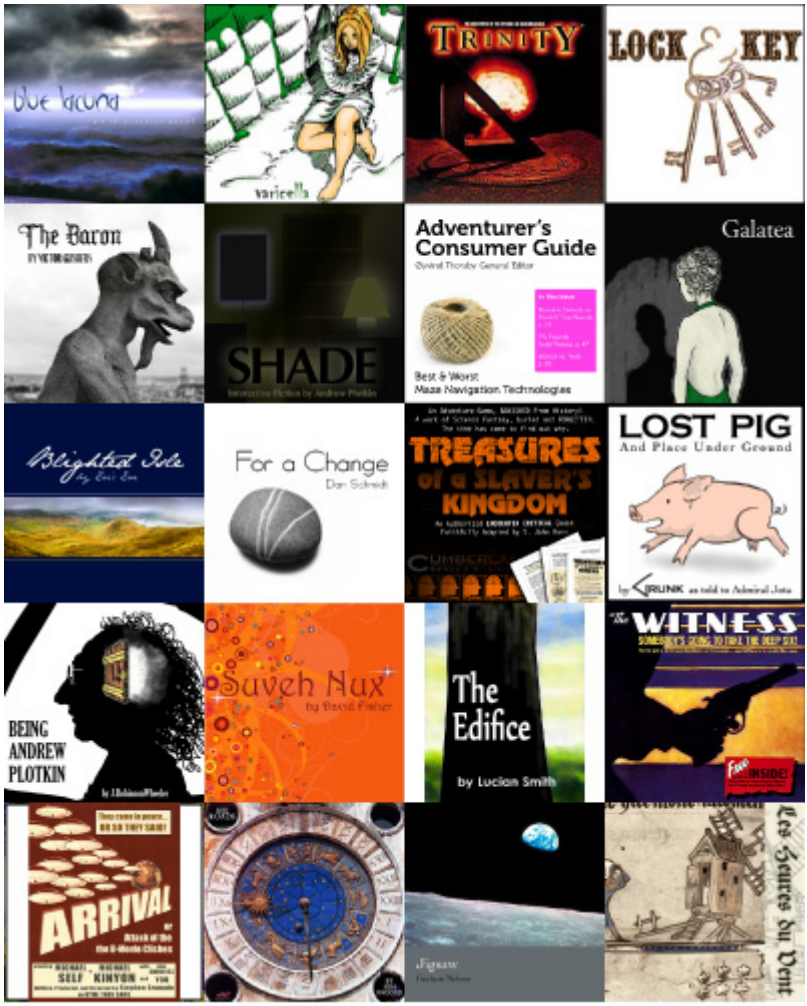
IF Archive; *§26.5. A Website of Its Own*; *§26.6. IFDB: The Interactive Fiction Database*; *§26.7. Compet*

Contents of *Writing with Inform*
Chapter 25: Releasing
Chapter 27: Extensions
Indexes of the examples

§26.1. Finding a readership

So the new work of IF is written, and tested, and has all its bibliographic data and a fancy cover illustration lined up. What next?

There is a thriving community of readers and writers of interactive fiction, and it is sometimes supported by grants from arts foundations and other cultural bodies: there's increasing attention from the academic world, and a general consensus has gradually grown that interactive fiction is a "valid" artistic medium for expression. Like poetry, it is something that a few people like a lot, and which most people can see the point of, even if they don't read it themselves. Over the last thirty years, a few authors have established durable reputations: they give occasional newspaper interviews, and have a very low-key kind of fame. There are competitions, and annual awards ceremonies. Newcomers are always welcome.



With some important exceptions, most works of interactive fiction have never been "published" in the sense of being issued for sale by a for-profit company. For the most part, IF has not been commercially valuable since about 1987. Successful authors of IF generally take the view that while they could, perhaps, make a very modest amount of money from sales, it would be a nuisance to collect and make no meaningful difference to their incomes; it would cut the number of readers, whereas one wants the satisfaction of being read; and besides, the whole culture of IF has always been characterised by giving and sharing. (Inform itself is free.)

Inform has nevertheless been used to produce commercial works (generally add-ons or bonuses to other games), and users are very welcome to sell works created by Inform with no royalty or requirement for rights clearance. It's also widely used in education, and as a prototyping tool for other kinds of story.

§26.2. How a novel is published

Suppose that a (traditional) novel has been delivered to a publisher: the author has handed it over as a pile of twenty chapters of prose, and feels that it is finished. In fact there is much still to do:

(a) Editing. An editor works through the book, looking for problems in the plot, uneven passages, difficulties of tone and a hundred other nuanced points. The author generally then revises the book and submits again.

(b) Copy-editing. A copy-editor fixes punctuation errors, awkwardly worded sentences and other low-level problems.

(c) Bibliographic data is added.

(d) Printing. The text is given a clean, readable rendition, and no longer looks like a home-made typescript.

(e) Cover art is added. Even unillustrated novels have pictorial covers, and these images are often used to set the tone for the book - they set the reader's frame of mind, so something more is happening than mere marketing.

(f) A back cover blurb is added. This will also find its way into catalogues, onto book trade databases, appear on Amazon.com and so forth. Both a description and a lure, it gives a flavour of the work without actually being any part of it.

(g) Binding. Not only are the inside pages printed, but maps, plates of illustrations, free CDs, fold-out charts, etc., may be tipped in to the binding.

(h) Legal deposit. Copies are lodged with libraries of record, such as the British Library or the Library of Congress, to ensure that the work cannot be lost from cultural history. (In most countries, this is a legal obligation for publishers.)

(i) Shipping. Copies are sent out to bookshops.

(j) Publicity. The author and publisher combine to put out the word, circulate leaflets, put up posters and so forth.

(k) Reviews and awards. Reviews are published, usually stirring up interest in the book. These having been stellar, a few months later the author bashfully accepts a

Pulitzer Prize, the Booker or some similar token of cultural esteem.

Start of Chapter 26: Publishing
Back to §26.1. Finding a readership
Onward to §26.3. How interactive fiction is published

§26.3. How interactive fiction is published

If we take the eleven novel-publishing stages of the previous section in order, we find that pretty well the same business goes on for works of IF.

(a) Editing. Working with a small number of trusted play-testers, and taking their responses seriously even when inconvenient, will almost always produce an immeasurably better work: not just better functionally, but better artistically, and more enjoyable. Play-testers can usually be recruited by placing an ad on **www.intfiction.org/forum**.

(b) Copy-editing. Play-testers will also pick up small stuff - spelling mistakes, and punctuation errors - but note that Inform for OS X will spell-check our source text on request.

(c) Bibliographic data is added.

(d) Printing. Clicking Inform's Release button is the equivalent the-die-is-cast moment.

(e) Cover art is added. As we saw in the previous chapter, Inform can add a cover image as part of the Release process, though it will not itself draw and design that image - like a printer, it expects to be supplied with the original.

(f) A back cover blurb is added. Inform does indeed allow us to compose such a piece of text and include it with the work's bibliographic data.

(g) Binding. The story file, which is akin to the inside pages of a book, is combined with its cover art, bibliographic data, and also with other non-textual materials provided by the author (booklets, sound samples, images, etc.). Inform does much of this automatically, producing a composite object called a "blorb".

(h) Legal deposit. The work is uploaded to the IF Archive (**www.if-archive.org**), whose librarians shelve it in the appropriate section. More on this later.

(i) Shipping. A work of IF is electronic rather than physical, so nothing is actually moved, but many authors like to put their works on their own websites as well as placing them in the Archive.

(j) Publicity. Authors often announce a new work on IFDB (**ifdb.tads.org**). Authors often also set up a personal web page about the work. Inform can generate such a web page automatically, as we saw in the chapter about releasing new works.

(k) Reviews and awards. The IF community has competitions and awards in abundance, and several websites gather reviews. It is usually safe to say that a well-written work will not go unnoticed if it is sensibly publicised.

Start of Chapter 26: Publishing

Back to §26.2. How a novel is published

Onward to §26.4. The IF Archive

§26.4. The IF Archive

Publishing an IF story consists of two steps. One is the technical task of making the story available to players - unless the plan is simply to email it to close friends, that means hosting it somewhere on the Internet. The second is the promotional task of letting people know the work exists, and where to find it.

It is a community tradition that all serious work is uploaded to the **IF Archive**, which is IF's answer to a national library. This is a mirrored, stable collection of thousands of interactive fiction games and programming languages, manuals, fanzines, maps, walkthroughs, and other materials. As such, it's likely to stay around even if a personal website goes off-line; it's also the primary resource for people doing scholarship on interactive fiction (and there are a growing number of these).

The Archive is very much a library, for long-term archiving, rather than a book-store. The catalogue is sober and textual, and there are no visual shop-windows, or posters advertising new titles hot off the press. Newcomers sometimes need practice finding their way around. And the Archive hosts story files (and associated manuals, as appropriate) but not advertising for them - it does not provide web-hosting for authors to set up mini-sites.

Uploading a work to the IF Archive is not too difficult, and can be done in two ways. One way is to use the archive's web form at:

http://www.ifarchive.org/cgi-bin/upload.py

The other is to create a new page at the Interactive Fiction Database, at:

http://ifdb.tads.org/

It's then possible to upload the story file to the IF Archive from IFDB. This is easiest all round, since it allows both IFDB and IF Archive to be updated at once.

In either approach, an author chooses and uploads a file, and accompanies it with a name and email address (so that the archive maintainers can verify the legitimacy of the work). The "About this file" field is for a line or two explaining what the story is -- its full title and any critical information -- and is used in generating the archive index. This is normally much shorter than the "blurb" described earlier. There's also a field to suggest where in the archive the story should be stored, but this is optional and intended chiefly for people expert in how the archive is filed. The archive maintainers will file a new story file in the obvious directory for its format. For Inform works, that means other Z-Machine - "z-code" - or Glulx story files. The maintainers sometimes place the same story file in multiple places in the Archive, using links.

As with all large libraries, it takes the Archive a little while for new acquisitions to be processed. When this happens, one of the volunteer maintainers will email with the official URL from which anyone can now download the story file.

Committing a story to the Archive is meant to be permanent. While the maintainers will happily replace older versions of stories with new improved releases, they are less eager to remove stories entirely. If that doesn't seem appealing, or if we do not want our story to be treated as freeware with essentially unlimited distribution, the Archive may not be a good choice. But it is deeply valued by the IF community, and has saved many works which could otherwise easily have been lost forever. Many contributions important in the history of IF were made by people who are now not easy to trace, and whose websites are long gone. But their work lives on.

Start of Chapter 26: Publishing

Back to §26.3. How interactive fiction is published

Onward to §26.5. A Website of Its Own

§26.5. A Website of Its Own

While any good story file ought to go into the IF Archive, it's probably wise also to provide an easier-to-use home for the work, by putting up a web page and a copy of the story file on a private web host. That host should ideally be as stable as possible, so that the URL is likely to remain fixed for what might be a long period. Freeware stories have a long period of viability relative to commercial games, which means that players may still be hearing about and checking out a story years after its initial release. A stable address helps everyone with links, and makes it easier for Google to direct people.

Of course creating a web page involves a little design work, but tools are widely available which make this quite easy nowadays. And as we've seen, Inform can automatically generate web pages and whole small mini-sites to put all the information about a story file into a tidy format, even including the ability to play online.

Start of Chapter 26: Publishing
Back to §26.4. The IF Archive
Onward to §26.6. IFDB: The Interactive Fiction Database

§26.6. IFDB: The Interactive Fiction Database

Once the story file has a home online, and a URL (that is, a web address) at which it can be found, it needs to be registered with IFDB:

http://ifdb.tads.org/

the Interactive Fiction Database. Just as the IF Archive is a repository for stories themselves, IFDB is a database containing information about them - titles, authors, locations, solutions, reviews, recommendation lists and more.

The name IFDB echoes the Internet Movie Database (IMDB), but in some ways it is also like the iTunes Music Store. For one thing, it's a shop-window for what's new, with cover art to catch the eye. For another, some interpreters allow players to browse IFDB directly and launch new stories in a single click. This kind of integration is only likely to increase, so story files unregistered on IFDB are likely to be much less visible to players of the future. Promoting IF is all about pulling in impulse players -- people who are passingly interested, but might not try the story if there is any significant work involved is setting it up. This is what IFDB is all about.

IFDB is community-editable, like Wikipedia, though editors are required to create an account and log in first -- this is free, of course. A standard form is provided for creating a new record (accessible by selecting the option to add a story listing). More or less the same information that appears on Inform's library card in the Contents index needs to be copied over: there's space for the author name, story title, genre, and so on. IFDB will also ask for an IFID, a code identifying the story uniquely. Inform generates one of these automatically for each project, and it, too, is on the Library Card. It can always be found by typing VERSION into the compiled story and looking at the line that says

Identification number: //[some letters and numbers]//

The part between the // marks is the IFID. If there's cover art, that can also be uploaded, and good cover art makes a big difference to shop-window-appeal.

The download link should give the most stable URL available. If you have not yet uploaded your story to the IF Archive, you may do so by selecting the "Upload it to the IF Archive" link instead of pressing the "Add a Link" button. The benefits of submitting your story to the IF Archive in this manner are two-fold. One, IFDB will fill in much of the information required by the IF Archive for you. Two, the link to your story will not appear until the IF Archive maintainers move it to its permanent home in the archive, at which point the download link will be automatically updated and presented on the story page.

If you choose to upload your story file to the IF Archive independent of IFDB, then once the story file is safely up at its permanent home on the IF Archive, that is an ideal address to quote here. Otherwise, the URL of the work's own website is best. (Note that the IFDB entry can always be edited later, if the URL moves.)

Commercial works which aren't available as free downloads can be registered on IFDB just the same, and this is almost certainly a good idea.

Start of Chapter 26: Publishing

Back to §26.5. A Website of Its Own

Onward to §26.7. Competitions

§26.7. Competitions

One very common way to get players for IF is to enter the story into an IF competition. The annual IF Competition is the most prestigious and has the widest field, but the Spring Thing, the One Room Game Competition and other events also catch people's attention. Entering a competition is a path of least effort for authors promoting their new work, because the competition organizer usually takes care of hosting and archiving submitted stories, promoting the competition as a whole, collecting votes, and encouraging players to post reviews. Different contests have different arrangements. The **ifwiki** usually posts a list of current and upcoming competitions, as well as lists of results for those recently past, on the front page:

http://www.ifwiki.org

Some competitions also have their own websites, at least at the relevant times of year.

All the same, there are many IF works that aren't cut out for competition release.

Competitions tend to be best for short or medium-short works, because judges don't necessarily have time to play a lot of long stories at once, and sometimes this is a condition of entry.

It's also good for publicity to win one of the annual **XYZZY Awards**. All interactive fiction stories released in a given year are eligible, and authors do not need to do anything to enter. As with the Oscars, though, you can't plan to win: it happens or it doesn't.

Start of Chapter 26: Publishing

Back to §26.6. IFDB: The Interactive Fiction Database

Onward to §26.8. The Gaming Avant-Garde

§26.8. The Gaming Avant-Garde

The IF community is not the only potential audience for a work of interactive fiction. Some authors have successfully written and pitched IF to other groups -- audiences interested in a particular historical period, role-playing story universe, or web comic, for instance -- and those groups have to be reached through their own community forums and meeting places.

Even if we do write material mostly meant for the existing IF community, that doesn't mean the audience has to stop there. There are a number of independent gaming websites and blogs that feature IF reviews occasionally or regularly. These things change quickly, but at the time of writing, IF is featured with some regularity on:

JayIsGames (www.jayisgames.com): a popular blog devoted to casual games, which occasionally profiles IF. JIG has also developed its own Flash interpreter for z-code games to allow the blog to host interactive fiction without requiring the casual audience to do any downloading. Experience suggests that JIG players are most interested in short IF -- serious or lighthearted, but humor goes over especially well -- and that they prefer very rigorously implemented work where a wide variety of player actions get customized replies. Small, polished gems do well here. JIG accepts story suggestions as well.

The Independent Gaming Source (www.tigsource.com): occasionally publishes reviews or shorter announcements, and has

even hosted an IF-writing competition. It may be worth drawing something to their attention.

This is hardly a complete list of gaming blogs with an interest in interactive fiction, just a collection of the most accessible ones, so it's worth doing a little research. One way is to pick a handful of works that coming closest to our own story in design and style, and search for reviews of those works. Where were they reviewed, and where were they well-received? Those venues might be good targets for our own production.

Finally, there is a constant round of competitions for independent games in general. Unlike IF community competitions, larger indie story comps usually don't require that a story be previously unreleased, only that it have no commercial funding. Some of these competitions offer substantial prizes in cash or computing equipment; some explicitly seek text-based games. Placing as a finalist in such a competition can mean having work displayed at a gaming expo or hosted on a special wesbsite, and this will garner substantial outside press. The field of competitors will be large and serious, but that doesn't mean it's not worth giving it a try.

This may all be a bit overwhelming. All the publicity options can seem like more work than we want to handle while simultaneously finishing a substantial opus. In fact, we don't have to do it all at once. One sensible approach is to release to the IF community first, and then approach the external websites later with a second or third release that puts a final polish on the story in response to player feedback. That process is even more important for indie gaming competitions. Having a release that's been polished, together with an attractive website, cover art, and maybe even some quotes from IF reviewers, can make a story look much more like a legitimate contender among semi-pro works.

Start of Chapter 26: Publishing
Back to §26.7. Competitions
Onward to §26.9. The Digital Literature Community

§26.9. The Digital Literature Community

If our work is more a work of digital literature than a game, different venues are appropriate. PlayThisThing may still be interested; JayIsGames probably will not be. Instead, we might want to look at

GrandTextAuto (http://grandtextauto.soe.ucsc.edu): a group blog about new media and interactive story-telling, which sometimes posts announcements of interactive fiction, especially of the more literary kind.

The Electronic Literature Organization (http://eliterature.org/): an organization dedicated to preserving all kinds of new media literature. They list a number of events (such as readings open to the public, new media gallery exhibits, etc.) that an IF author could participate in, and they also have a directory of electronic literature, to which we can add our own works.

Digital Humanities Quarterly (http://www.digitalhumanities.org/dhq/): A scholarly publication, but one that looks at all aspects of digital literature, new media tie-ins for scholarship, etc. DHQ has run several articles about interactive fiction in the past, including Dennis Jerz's ground-breaking research on the writing of the first text adventure, Colossal Cave. DHQ might be open either to hosting a work or to publishing an article about it.

That still leaves out a category of IF -- namely, work written for a popular reading audience, work that isn't trying to be literary but also isn't primarily game-like. There are, at the moment, no ideal venues for promoting such work (that we're aware of), but it's certainly worth pursuing.

Start of Chapter 26: Publishing
Back to §26.8. The Gaming Avant-Garde
Onward to §26.10. A short concluding homily

§26.10. A short concluding homily

It's natural to want to make a huge splash with a story, but in the IF community, instant widespread adulation for any work is pretty uncommon.

For one thing, players tend to play when they get around to it... which may be weeks, months, or even years after the initial release. Reviews trickle rather than flooding in. Appreciation builds slowly. And sometimes works that placed unspectacularly in a competition, or seemed to be overlooked in the annual XYZZY Awards, gradually come to be regarded as classics because of some pioneering technique.

So it's wise (if difficult) not to judge a story's success entirely by its immediate feedback. Even after its debut, a story can often use a little care and attention if it's to reach all its potential fans -- whether that means building further releases, posting hint files or walkthroughs, developing new websites, or approaching outside reviewers.

Start of Chapter 26: Publishing
Back to §26.9. The Digital Literature Community
Onward to Chapter 27: Extensions: §27.1. The status of extensions

Chapter 27: Extensions

nform 7; *§27.15. Defining phrases in Inform 6*; *§27.16. Phrases to decide in Inform 6*; *§27.17. Handling*

Contents of *Writing with Inform*
Chapter 26: Publishing
Indexes of the examples

§27.1. The status of extensions

The range of simulation offered by Inform's model world is intentionally limited to a core of basic essentials. We could argue at the margins, and the choice of what's in and what's out is partly traditional, but most people find the model reasonable as far as it goes.

Between 1993 and 2006, quite a range of "library extensions" for Inform 6 was written. Most of these extensions aimed to fill out the model by simulating other aspects of life, too: money, clothing, pourable liquids. None of these extensions was official and all of them were: it was a free-for-all, and in several cases different authors wrote rival extensions to model the same basic ideas. The development of Inform 7 was strongly influenced by this history and by the recognition that the base of rules and grammar inside a typical modern story are seldom written by a single author. They combine the standard Inform material with extensions by several third parties, together with anything specific to the story in question.

Inform 7 has a more organised idea of extensions, as we shall see. But anyone is free to write an extension on any terms or for any reason. Writers may wish to use the techniques in this chapter to develop private extensions of their own, used in several projects, or to share them with associates but not more widely.

But most writers of extensions do so to contribute to the Inform community, and for the satisfaction of solving a problem. Inform does not recognise anyone's approach to a particular need as "the official solution" - for instance, although the standard Inform distribution includes a copy of Locksmith by Emily Short, that is not the "official" way to make automatically unlocking doors, and anyone is welcome to try a better one.

However, the Inform project does recognise some extensions as "public". Public extensions are the ones archived on the Inform website for the free use of all Inform writers. Those who wish to contribute an extension as a public one are obliged to follow a number of guidelines, which are mostly stylistic points intended to make the range of extensions easier to work with. Extension writers are asked to join in the spirit of these rules and help make the whole cooperative enterprise work harmoniously. Extensions which do play by these rules are also accepted into the

Public Library, which makes them easy for all Inform users everywhere to find and obtain them.

Writers who wish to make their extensions public on the Inform website should also be clear that by doing so, they are donating their work to the community on the basis of the broadest form of Creative Commons license: that is, they retain copyright and the right to be identified as the author (and as we shall see they are automatically credited in any work of IF which uses their extension), but are giving unlimited permission to use, circulate and republish their extensions in any form, even as part of commercial works (should that arise). To publish a public extension is a public-spirited act, done for only the reward of a modest acknowledgement.

If the author of an extension has not made it public, or indicated in some other way that it is free to be used without the need for permission, then it would be both polite and prudent to check with the author before publishing something which incorporates his work.

Start of Chapter 27: Extensions

Back to Chapter 26: Publishing: §26.10. A short concluding homily

Onward to §27.2. The Standard Rules

§27.2. The Standard Rules

When any source text is run through Inform, a secret first line is inserted, which reads:

```
Include the Standard Rules by Graham Nelson.
```

The "Standard Rules" file contains the definitions of the basic kinds, phrases, actions and grammar described in this documentation: for instance, it includes lines like

```
A container is a kind of thing.
```

...without which Inform would be lost. Although including the Standard Rules is compulsory, it is treated internally as if it were any other "extension".

What happens when an "Include" sentence is reached is that the sentence is replaced with the whole text of the file in question, often many paragraphs long.

If the file has already been included, then the sentence is simply ignored. This is so that we can have two extensions, each of which needs the other: if A says to include B, and B says to include A, the result is that including one automatically includes the other, so we always get both which ever we ask for - not that there is a hideous infinite regress.

Start of Chapter 27: Extensions

Back to §27.1. The status of extensions

Onward to §27.3. Built-in, installed and project-specific extensions

§27.3. Built-in, installed and project-specific extensions

To recap: Inform builds projects from both the source text typed by the author and from Extensions; one of these, the Standard Rules, is always included; others are added as authors please. About 20 are "built-in" to Inform, meaning that they are stored inside the application and always available. Others must be "installed", and each Inform user will have a folder somewhere on his computer which contains these. Users typically obtain these from the Public Library feature in the application, but they can also download them directly from inform7.com and then use an Install Extension menu option in the application. Either way, the application then squirrels the file away, and it becomes available to any projects that that user may be working on.

It is also possible to have extensions available to just one project. These must be stored in the Extensions subfolder of the project's ".materials" folder, but otherwise are arranged the same as installed extensions - there's an outer folder for each author's name, and extensions are named with a ".i7x" extension within. For example:

```
Mourning Hypercritical.inform
Mourning Hypercritical.materials
    Extensions
        John Siracusa
            Fixing The Finder.i7x
```

When Inform needs to find an extension, it looks here first, then in the installed area, then in its built-in area. That means that we can make our own revised or hacked version of an extension, put it in the ".materials" area, and then have it take precedence over the installed or built-in one. We could even have our own private version of the Standard Rules here.

(This has a number of possible uses - for example, to provide a convenient test-bed

when working on an experimental version of an extension.)

Start of Chapter 27: Extensions

Back to §27.2. The Standard Rules

Onward to §27.4. Authorship

§27.4. Authorship

Extensions are identified by author and by name, so that a given author can produce his or her own range of extensions, and need only ensure that these are named differently from each other. If John Smith and Mary Brown each want to write an extension called "Following People", there is no conflict.

The name of an extension, and of an author, should be written in Sentence Capitalisation: that is, upper case for the first letter in each word. (Inform uses this to minimise problems on machines where filenames are read with case sensitivity.) It is permitted for author names to include upper-case letters within words, as with the "G" in "Jesse McGrew". In general it is best to avoid accented or unusual letters in titles and author names, but the standard ISO Latin-1 characters should be allowed - for instance,

Étude Pour La Fênetre by Françoise Gauß begins here.

The author name must not start with "The", nor contain the words "by", "and" or "version", or contain punctuation, as in "John X. Doe"; the title similarly, except that "and" is permitted. Name and author's name must each be no more than 50 characters long, including any spaces between words.

Authors are asked to use real names rather than cryptic handles like "ifguy", and to use genteel, plausible pseudonyms like "Emily Short" rather than, say, "Drooling Zombie" or "Team Inform". Authors are also asked to use the same author's name for all their own extensions, and (it should go without saying) not to masquerade as anybody else.

Sometimes authorship is complicated. What if Mary Brown finds some Inform 6 code written by John Smith in the mid-90s, and puts an I7 gloss on it to make an I7 extension, but then Pierre Dupont translates it into French: who's the author of the result? The rule is that the person making the current, latest version is the author listed in the titling line, so we end up with

```
... by Pierre Dupont begins here.
```

But Mary and John deserve their credits too: see the next section for how to give them.

Start of Chapter 27: Extensions

Back to §27.3. Built-in, installed and project-specific extensions

Onward to §27.5. A simple example extension

§27.5. A simple example extension

Extensions are plain text files, and can be created with any text editor. (It is sometimes said that "there is no such thing as plain text", there being so many ways to represent exotic characters: so to be precise, an extension is a text file with the Unicode UTF-8 encoding, either with or without a BOM marker, using any of the possible forms of line-ending (Unix, Windows, Macintosh, or Unicode line divider). This is a detail which will only matter if the extension contains accented letters or other exotica.)

Extensions look very much like passages of Inform source, because except for a special introductory and concluding sentence, and one convention, that is all they are:

```
The Ducking Action by Graham Nelson begins here.

"An action for ducking one's head."

Ducking is an action applying to nothing. Report ducking: say "You
duck!" Understand "duck" as ducking.

The Ducking Action ends here.
```

Not a useful or interesting extension, but those few words add a whole new action and everything needed to make it work. It is Inform's ability to mix up rooms, things, kinds, grammar, phrases and rules, in more or less any order, which makes it

possible for extensions to work.

The introductory sentence must be placed as the only content of line 1 of the file, which must not contain comments, and has to be written in exactly the correct form. Inform checks this very carefully when performing its census of installed extensions, on each translation of the text. (In case the extension's title is a plural, we are allowed to write "begin" and "end" instead of "begins" and "ends". For instance, the last line of the standard rules is "The Standard Rules end here.")

The "one convention" mentioned above is that if a double-quoted text is placed immediately after the beginning sentence (and with no intervening comments), then it is taken to be a short description of the extension's content called the "rubric". Hence the line:

```
"An action for ducking one's head."
```

Providing a rubric is helpful, because it enables Inform to give a meaningful listing even for an as-yet unused and unindexed extension, and because it helps the Inform website to produce better directories. Note the word "short": such text is likely to be truncated if it exceeds 500 characters.

A second double-quoted text can also, optionally, be added in yet a third special starting paragraph. This is to provide additional credits to people who have contributed to this or earlier versions. For instance:

```
The Ducking Action by Graham Nelson begins here.

"An action for ducking one's head."

"based on original Inform 6 code by Marc Canard"
```

Note the typical style here: it's a phrase rather than a sentence, and neither starts with an upper-case letter nor ends with a full stop. (The additional credit is then used in documentation and also in the VERSION text of any Inform story file using the extension.)

Onward to §27.6. Version numbering

Example 449: ★★ **Modern Conveniences** Exemplifying the kind of source we might use in writing extensions for kitchen and bathroom appliances.

---

§27.6. Version numbering

As we have seen, extensions are referred to by name and author, but they can also (optionally) be referred to by version. For instance:

```
Include version 2 of the Ducking Action by Graham Nelson.
```

```
Version 1/040426 of the Ducking Action by Graham Nelson begins
here.
```

A version number must be a whole number 1, 2, ..., 999 (version 0 is not allowed, and nor are decimal points), optionally followed by a slash '/' and then a six-figure date, in the form YYMMDD: so 040426 means 26 April 2004.

A request to include version 2 of something implicitly means "version 2 or later", and similarly "version 2/040426" means version 2 with a date of 26 April 2004 or later, or else any version higher than 2. If Inform loads the extension but finds that its version is not good enough, an error is issued.

Where no version number is quoted, the rule is that an unspecified version predates any numbered version. This means that the line

```
Include the Ducking Action by Graham Nelson.
```

will be happy with any version of the extension at all, whether numbered or not; but

```
Include version 2 of the Ducking Action by Graham Nelson.
```

will only accept the extension if it has a version number attached (and with that

number being 2 or higher).

During play of any story compiled by Inform 7, typing VERSION lists various serial numbers of the pieces of software used to make it. The list concludes with names, authors and version numbers of any extensions used. So every author whose work contributes to a story automatically gets a modest credit within it. The same list can be printed, at the discretion of the designer, using the textual substitution:

say "[the/-- list of extension credits]"

This text substitution expands to one or more lines of text crediting each of the extensions used by the current source text, along with their version numbers and authors. Extensions whose authors have chosen the "use authorial modesty" option are missed out. Example:

```
Standard Rules version 2/090402 by Graham Nelson
```

If we want our extension to go uncredited - perhaps if it is a low-level enabling sort of thing, for instance - we can place the following sentence inside the definition of the extension:

```
Use authorial modesty.
```

The same sentence placed in the body of a source text causes all extensions by the same author as the main source text to go uncredited. In other words, if Isaac Miggins writes a source text and includes, say, Unlikely Events by Isaac Miggins, then this extension will go uncredited in the VERSION command.

A complete list, undiluted by modesty, can always be obtained using:

say "[the/-- complete list of extension credits]"

This text substitution expands to one or more lines of text crediting each of the extensions used by the current source text, along with their version numbers and authors. Every extension is included, even those whose authors have opted for "use authorial modesty". Example:

```
Standard Rules version 2/090402 by Graham Nelson
Locksmith version 9 by Emily Short
```

§27.7. Extensions and story file formats

Inform compiles to several different story file formats, and in each case uses only a small part of their abilities - especially when it comes to fancy tricks with the keyboard or screen. So people may well want to write extensions which provide access to some of these tricks (like "Basic Screen Effects", included in the standard Inform distribution, but more so). Unfortunately, these tricks are very likely to fail to compile - or fail to work - on some of the possible story file formats, so the resulting extension would probably go wrong (and mysteriously wrong) for users who have chosen a different format.

Inform therefore provides a way for extensions to declare the formats they are compatible with. All that is required is to add a proviso in brackets after the title is declared:

```
Version 2 of Basic Screen Effects (for Z-Machine version 8 only) by
Emily Short begins here.
```

Other examples might be "(for Glulx only)", or "(for Z-machine only)". If no such proviso is given, the extension is assumed to be compatible with every story file format.

Extensions are also able to include material which is only used on some story file formats and not others - in principle, this might allow the same facilities to be provided to the author whatever story file format is used, but to achieve these effects differently depending on the current Settings. The convention here is exactly like "not for release": if a heading or subheading in the source text contains a bracketed proviso, then the material under that heading (and under its dependent subheadings) will be ignored if the current story file format does not match. For example:

```
Section 2.3G (for Glulx only)
```

```
To reveal the explosion:
    [...the Glulx way...]
```

```
Section 2.3Z (for Z-machine only)
```

```
To reveal the explosion:
    [...the Z-machine way...]
```

would ensure that "reveal the explosion" works nicely whichever story file format is used.

Start of Chapter 27: Extensions

Back to §27.6. Version numbering

Onward to §27.8. Extensions can include other extensions

Example 450: ★★ **Tilt 3** Displaying the card suits from our deck of cards with red and black colored unicode symbols.

§27.8. Extensions can include other extensions

As was mentioned earlier, the same extension is sometimes requested several times. For instance, suppose the main source text asks to include version 2 of extension X, and also to include extension Y. Suppose further that Y contains a request to include version 4 of X. We now have two different requests for X. Inform works out the minimum version number needed to satisfy these requests (in this case 4) and gives an error if the extension it actually loads turns out to be earlier.

This is true even of the Standard Rules. Suppose that the Standard Rules had recently been republished in version 37, making the previously existing version 36 out of date, and that an extension is written which capitalizes on a new feature in v37. It will therefore not work if people try to use it with v36. All the extension needs to do is to say:

```
Include version 37 of the Standard Rules by Graham Nelson.
```

to guarantee that v37 or later will be used.

If an extension does include other extensions, it should do so in a paragraph immediately following the introductory sentence, so that anyone looking at the file can see these mutual dependencies at a glance.

Start of Chapter 27: Extensions

Back to §27.7. Extensions and story file formats

Onward to §27.9. Extensions can interact with other extensions

§27.9. Extensions can interact with other extensions

When one extension is being used, it's probably only one among several. A really general-purpose extension might want to behave differently depending on which other extensions are also present. This can be achieved using headings which are "for use with" (or "without") other extensions. For instance:

```
Chapter 2a (for use with Locksmith by Emily Short)
```

specifies that everything under this heading (and its subheadings, if any) will be ignored unless the extension Locksmith by Emily Short is included. Conversely,

```
Chapter 2b (for use without Locksmith by Emily Short)
```

will be ignored unless it isn't included. This allows an extension to give two variations on the same material - one if Locksmith is present, the other if not.

Headings can also replace portions of extensions which have been included. For instance:

```
Section 6 - Hacked locking (in place of Section 1 - Regular locking in
Locksmith by Emily Short)
```

places the source text under the new heading in the place of the old (which is thrown away). If there should be two or more headings of the same name in the given extension, the first is the one replaced; if two or more headings attempt to replace the same heading in the given extension, the final attempt in source text order is the one

which succeeds; and finally, heading dependencies like the above are scanned in a top-down way. Thus, if we have:

```
Chapter 2a (for use with Locksmith by Emily Short)
...
Section 1 - Hacked marbles (in place of Section 4 in Marbles by
Peter Wong)
...
```

and we don't include Locksmith, then the replacement of Section 4 of Marbles is not made, because Section 1 - Hacked marbles is subordinate to the Chapter 2a heading which we've told Inform to ignore.

If the name of the heading to replace contains the word "in", it's a good idea to use quotation marks for clarity:

```
Section - Hacked questions (in place of "Section 4 - Phrase used to
ask questions in closed mode" in Questions by Michael Callaghan)
```

Start of Chapter 27: Extensions

Back to §27.8. Extensions can include other extensions

Onward to §27.10. Extensions in the Index

§27.10. Extensions in the Index

As soon as a project has successfully been translated, its Index is brought up to date: pages of the index record all the kinds and what they are for, all the phrases which can be used, and so on. Any kind or phrase created in an extension is automatically included. The extension's presence in the project is itself recorded - the Contents index for any project contains a brief list of all extensions used in that project, along with their authors and version numbers. For instance:

```
Standard Rules version 1/040731
Locksmith by Emily Short
Tupperware by Emily Short
```

The Kinds index aims to give the reader a brief note of what each kind is intended for.

We can provide for this by writing a sentence like so:

```
The specification of player's holdall is "Represents a container which
the player can carry around as a sort of rucksack, into which spare
items are automatically stowed away."
```

There is no need to specify the properties which apply: that is all done automatically. "Specification" is a sort of pseudo-property used just for this: we can also give specifications to kinds of value and to actions, and these are similarly used in the Index pages.

Every extension has the right to its own set of headings and subheadings, independently of those used by the main source for the work or by any other extension which may be included. (So if the extension is divided into four sections and finishes on Section D, say, that doesn't mean that Section D will continue outside the extension as the main source of the story runs on.)

Extensions should, of course, be written so that they never produce Problem messages, so at first sight it appears that these headings will never be outwardly visible. In fact, though, Problems do occasionally turn up in extensions, usually when the user has made a mistake, or when two inconsistent extensions are used in the same project. But more importantly, the headings in an extension are used when indexing phrases (and also actions) to group similar phrases together. For instance, the Standard Rules contain the heading:

```
Section SR4/7 - Searching and sorting tables
```

The half-dozen phrases defined in this section of the Standard Rules are then indexed under the subheading "Searching and sorting tables": Inform looks for a hyphen in the heading and then uses any text which follows the hyphen. (If there is no hyphen, the entire heading text is used.)

If an extension contains no headings, its phrases (or actions) are indexed simply as "Miscellaneous".

Finally, any phrase or variable defined immediately under a heading whose name ends in the word "unindexed" will be omitted from the Phrasebook or Contents index respectively. (That won't apply to definitions under subheadings of the heading.) This is intended so that technical apparatus used only inside the extensions can be concealed from the outside user's immediate view. Inform as it is presently constituted does not allow extensions to make fully private definitions, but this feature at least allows them to make unadvertised ones.

Back to §27.9. Extensions can interact with other extensions
Onward to §27.11. Extension documentation

§27.11. Extension documentation

A basic mechanism for documenting extensions is built into Inform. For many extensions, this will probably do instead of a manual; for more complex ones, it should still prove a useful supplement to one.

As described in Chapter 2 above, whenever an extension is installed, its documentation is made available to the user. Such text should be written concisely, while giving examples wherever appropriate. Stylistically, it should ideally follow the model of the main Inform documentation: just as an extension expands the standard rules, so its documentation expands this manual. "We need..." is preferred to "You need...", and so on: we're all in this together.

In order to be recognised as documentation, this text should appear at the foot of the extension file, *after* the compulsory end sentence. The first paragraph must have exactly the following form, with a skipped line before and after:

```
---- DOCUMENTATION ----
```

For instance, the "Ducking Action" example might end:

```
...
The Ducking Action ends here.

---- DOCUMENTATION ----

This is a modest extension, with much to be modest about. It allows
us to use a new action for ducking, as in ducking the player's head
(not as in ducking a witch). Ducking will do nothing unless rules are
added:
```

```
Instead of ducking in the Shooting Gallery, say "Too late!"

...
```

We obtain indented code examples by beginning a line with a tab. A double indentation can be got with two tabs in a row, and so forth. (Beware: some text editors, or emailers, flatten tabs into a row of four or perhaps eight spaces each. Inform will not recognise such a line of spaces as a tab.)

Note that text in square brackets should be avoided in the documentation, because that's taken as being comment matter on the extension, and omitted.

Tables should be similarly indented, and should begin with the word "Table ...": the top line is taken to be the name of the table, and subsequent lines are tab-divided columns. Inform will automatically group this into a table, like so:

```
Table of Exemplariness

example object
galaxy "Andromeda Galaxy M31"
star "Sirius"
planet "Neptune"
moon "Europa"
dwarf planet "Quaoar"
comet "C/1983 H1 (IRAS)"
asteroid "(38628) Huya"
thing "Easterbunny"
```

(Footnote: Since the first appearance of this book, Easterbunny has been renamed Makemake, the creator god in the mythology of the people of Easter Island.)

Start of Chapter 27: Extensions
Back to §27.10. Extensions in the Index
Onward to §27.12. Examples and headings in extension documentation

§27.12. Examples and headings in extension documentation

Extensions with very large amounts of documentation can, if the author chooses,

divide the material up using headings and/or subheadings. These must be written as paragraphs exactly like so:

```
Chapter: Avoiding Events

Section: Ducking examinations and tests
```

Inform will then typeset them to stand out, will number them automatically, and will add a table of contents at the top of the page. (For most extensions, the documentation will be short and sweet, and this would just be clutter: headings and subheadings are best used only where the text would otherwise be difficult to read.)

Any extension's documentation can contain Examples, just as the main Inform documentation does: these are automatically labelled A, B, C, ... rather than given numbers, to ensure that they do not clash with the numbering used in the built-in chapters. (The labels may be helpful in writing an extension's documentation: we can write, for instance, a note such as "see Example C below".)

Examples must be given last in the documentation, and there can be up to 26 of them, though most extensions will need one example at the most, and some will have none at all. Each example must begin with a paragraph exactly like so:

```
Example: ** We Must Perform a Quirkafleeg - Ducking to avoid
arrows as one proceeds east across battlements.
```

Again, there must be a skipped line before and after. The row of asterisks must be *, **, *** or ****, just as in the main documentation, which we should follow on all points of style. The rest of the line contains the title, a hyphen, and then the description. The title should be given with Each Word except Prepositions and Similar Things Capitalized, while the description should look like a sentence, and end with a full stop.

The text of the example follows, of course, and continues until the end of the file, or the next "Example:" line, whichever comes first.

Each example should (normally) contain one single, complete, story, long enough to demonstrate the use of the extension and to have a little flavour to it, but not so long that the reader gets lost. It should have a title, which should match the name of the example (in the case above, "We Must Perform a Quirkafleeg"). It should conclude with a paragraph defining a test:

```
Test me with "east / duck / east / jump / east / duck / east / rescue
```

```
esmerelda".
```

The idea is that typing one single command, TEST ME, into the resulting story should show off what the extension does.

When an extension contains more than one example, they should be given in order of asterisk rating, that is, starting with the * examples, then the ** examples, and so on up.

Start of Chapter 27: Extensions

Back to §27.11. Extension documentation

Onward to §27.13. Implications

§27.13. Implications

Extensions often need to define new kinds or properties, which we want to make as helpful as possible for the user. In particular, we want them not to require additional work for the author just to obtain the effect which seems only natural.

For example, consider Inform's built-in "locked" property. If a door is locked, then it cannot be opened, which seems fair enough. But if the player tries to unlock the door, he might then find the following response:

```
That doesn't seem to be something you can unlock.
```

Which does not seem right. In real life, almost all locked items have outwardly exposed locks which it is perfectly sensible to try to unlock, given a key. The problem is that our door has the "locked" property, but not the "lockable" one.

The Standard Rules solve this problem by including the following line:

```
Something locked is usually lockable.
```

This ensures that any door said by the author only to be "locked" will be "lockable" as well, and adds a small but worthwhile touch of realism.

Such a sentence is called an "implication", as it is in the form "Condition A implies Condition B". Note that the two conditions must consist of either/or properties with or without kinds attached. Thus:

```
A room in the Open Desert is usually lighted.
```

will not work because "a room in the Open Desert" is a more complicated grammatical construction than, say, "lighted" or "a lighted room": it contains a relative clause. Inform can only deal with simple implications.

Inform never overrides certainties with mere implications, and is cautious about allowing them to build overly long chains of argument. This is to prevent the following kind of difficulty:

```
An open door is usually closed. A closed door is usually open.
```

Implications work just the same for values which aren't objects, so:

```
Colour is a kind of value. The colours are red, green and blue.
A colour can be zesty or flat. A colour can be bright or dull.
Red and blue are bright. Blue is flat.

A bright colour is usually zesty.
```

results in red being zesty, but blue and green being flat; blue because the source text explicitly says so (which trumps the "usually"), and green because this isn't a bright colour, so the implication doesn't arise.

Implications have not been mentioned up to now since they are only really needed by extensions, but also because they can be tricky, with unforeseen consequences. We should handle them with care.

Start of Chapter 27: Extensions

Back to §27.12. Examples and headings in extension documentation

Onward to §27.14. Using Inform 6 within Inform 7

§27.14. Using Inform 6 within Inform 7

To return to what was said on the first page of this documentation: Inform 7 (or I7), the current version, is not at all like Inform 6 (or I6). Internally, however, I7 works by translating the source text into an I6 program, so that in a practical sense the current version of Inform actually contains its predecessor.

The final sections of this chapter show how I6 code can be mixed directly in with I7. The remaining pages will therefore make little or no sense to those who do not already use I6. But for those who do know I6 already, it would be all too easy to write highly hybridised code, constantly mixing I6 and I7. The authors of Inform hope that this will not happen: for almost all purposes, I7 is much more powerful than I6, and fails - when it has to fail - in a way more helpful to the user. Ideally, all I6 content would be confined to extensions (and this may be mandated in future releases of Inform), and even writers of extensions are asked to pare down their usage of I6 to the minimum necessary.

The methods for incorporating I6 code into I7 have been designed with this in mind, that is, to encourage people to use I6 in as self-contained a way as possible: in particular to isolate the relatively few functions which need to be written in I6, and to give them natural language expression.

Finally, anyone hacking with I7 for a while is likely to become curious about the Standard Rules file, and to look at the text on which the Inform world model is founded. The file is, of course, no secret, but it can be misleading. For one thing, it appears to have great freedom to set up the world model as it pleases, but in fact the I7 compiler may well crash unless certain things are done just so in the Standard Rules: they depend on each other.

Moreover, the Standard Rules use a number of syntaxes which are not documented in this chapter: these are constantly being altered, and it would not be safe to imitate them. Any I6-related syntax which is not documented in this chapter may be removed or changed in effect at any time without warning, for instance in an update of Inform to fix bugs.

Start of Chapter 27: Extensions

Back to §27.13. Implications

Onward to §27.15. Defining phrases in Inform 6

§27.15. Defining phrases in Inform 6

The phrases described in this documentation, such as "end the story", are all defined in the Standard Rules, and are for the most part defined not in terms of other I7 phrases but instead reduced to equivalents in I6. For instance:

```
To end the story (- deadflag=3; story_complete=false; -).
```

The notation "(-" and "-)" indicates that what comes in between is I6 code. The minus sign is supposed to be a mnemonic for the decrease from 7 to 6: later we shall use "(+" and "+)" to go back up the other way, from 6 to 7.

When a phrase is defined as containing only a single command, and that command is defined using I6 - as here - it is compiled in-line. This means that the phrase "end the story" will always be translated as "deadflag=3; story_complete=false;", rather than being translated into a call to a suitable function whose only statement is "deadflag=3; story_complete=false;".

This is an easy case since the wording never varies. More typical examples would be:

```
To say (something - number): (- print {something}; -).
To sort (T - table name) in (TC - table column) order:
    (- TableSort({T}, {TC}, 1); -).
```

When the braced name of one of the variables in the phrase preamble appears, this is compiled to the corresponding I6 expression at the relevant position in the I6 code. So, for instance,

```
say the capacity of the basket
```

might be compiled to

```
print O17_basket.capacity;
```

because "{something}" is expanded to "capacity of the basket" (I7 code) and then translated to "O17_basket.capacity" (I6 code), which is then spliced into the original I6 definition "print {something};".

Braces "{" are of course significant in I6. A real brace can be obtained by making the character following it a space, and then I7 will not attempt to read it as a request for substitution.

It's also possible for the pair of characters "-)" to occur in I6 code, for example here:

```
for (i=3 : i>0 : i--)
```

and I7 will read the "-)" as terminating the I6; we can get around this with an extra

space:

```
for (i=3 : i>0 : i-- )
```

Warning: Inform 6 uses a restricted character set, allowing use of most of the accented characters in ISO Latin-1 (those found in a set called ZSCII) but little beyond that. It's therefore hazardous to use any exotic Unicode characters in an inclusion.

Start of Chapter 27: Extensions

Back to §27.14. Using Inform 6 within Inform 7

Onward to §27.16. Phrases to decide in Inform 6

Example 451: ★★★ **Pink or Blue** Asking the player to select a gender to begin play.

§27.16. Phrases to decide in Inform 6

There are basically three forms of phrase in I7: phrases which do something, but produce no value or opinion as a result; phrases to decide whether or not something is true; and phrases to decide on a value. We have already seen examples of writing the first form in I6:

```
To say (something - number): (- print {something}; -).
```

Here the I6 form is required to be I6 routine code in void context, that is, it will normally be one or more statements each of which ends in a semicolon (unless there are braced code blocks present). In this case, we have just one I6 statement, ending in a semicolon.

An example of a phrase to decide whether something is true would be:

```
To decide whether in darkness: (- (location==thedark) -).
```

Here the I6 code providing the definition must be a valid I6 condition, and be in round brackets, but there is no semicolon.

Lastly, an example of a phrase to decide on a value:

```
To decide which number is the hours part of (t - time): (- ({t}/60) -).
```

Again, this is a value in I6 as well: no semicolon. It is probably safest to place the value in round brackets.

Start of Chapter 27: Extensions

Back to §27.15. Defining phrases in Inform 6

Onward to §27.17. Handling phrase options

§27.17. Handling phrase options

The Standard Rules use the Inform list-writer with the following definition, which shows how a much more complicated I6 routine can be given a natural-language expression.

```
To list the contents of (O - an object),
    with newlines,
    indented,
    giving inventory information,
    as a sentence,
    including contents,
    including all contents,
    tersely,
    giving brief inventory information,
    using the definite article,
    listing marked items only,
    prefacing with is/are,
    not listing concealed items,
    suppressing all articles
    and/or with extra indentation:
    (- I7WriteListFrom(child({O}), {phrase options}); -).
```

This can be used by, say:

```
list the contents of O, as a sentence, using the definite article
```

"{phrase options}" is a special substitution: it is a bitmap which assigns the given options one bit each, starting with the least significant bit for the first-mentioned option ("with newlines" above) and going up to the most significant bit for the last ("with extra indentation").

Start of Chapter 27: Extensions
Back to §27.16. Phrases to decide in Inform 6
Onward to §27.18. Making and testing use options

---

§27.18. Making and testing use options

Use options (see Chapter 2 above) manifest themselves in the I6 code generated by I7 as constants which are either defined, or not. For instance, the "use American dialect" option results in the constant DIALECT_US being defined, a constant which otherwise would not be. Some use options define the constant as a particular value, others simply define it (so that I6 gives this constant the value 0).

New use options can be created as in the following examples, which are found in the Standard Rules:

```
Use American dialect translates as (- Constant DIALECT_US; -).
Use full-length room descriptions translates as (- Constant
I7_LOOKMODE = 2; -).
```

Most Inform users will not need to test whether a use option is currently set: after all, they will know whether or not their own story uses American dialect. But an extension does not know what use options apply in the story which is using it. An extension which needs to print a list, using its own formatting, might want to know whether "use serial comma" is set. Or it might want to speak differently in American dialect.

To test for American dialect, we should ideally not use I6 to look for the constant DIALECT_US using #ifdef: there is no guarantee that this constant will not be renamed at some point. Instead we can perform the test directly in I7:

```
if the American dialect option is active, ...
```

and similarly for all other named use options. The adjectives "active" and "inactive" have the obvious meanings for use options. This means it's possible to describe the current options like so:

```
say "We're currently using: [list of active use options].";
```

The result might be, say,

```
We're currently using: dynamic memory allocation option [8192],
maximum text length option [1024], maximum things understood at
once option [100], American dialect option and fast route-finding
option.
```

This may be useful for testing purposes.

Use options can also allow the writer to raise certain maximum values. If we write an extension which needs some I6 array, say, and therefore has some limitation - for instance a footnotes presenter which can handle at most 100 footnotes before its array space runs out - it would obviously be cleaner to allow this maximum to be raised. We can set this up like so:

```
Use maximum presented footnotes of at least 100 translates as (-
Constant MAX_PRESENTED_FOOTNOTES = {N}; -).
```

With such a definition, the number given is the default value, and the I6 source is included whether or not anybody uses the option: the default value being given if nobody does. The text "{N}" is replaced with the value. So the above definition normally results in this being defined:

```
Constant MAX_PRESENTED_FOOTNOTES = 100;
```

but if the user writes

```
Use maximum presented footnotes of at least 350.
```

then instead the I6 inclusion becomes:

```
Constant MAX_PRESENTED_FOOTNOTES = 350;
```

The I6 constant MAX_PRESENTED_FOOTNOTES can then be used as the size of

an array, for instance.

Start of Chapter 27: Extensions
Back to §27.17. Handling phrase options
Onward to §27.19. Longer extracts of Inform 6 code

§27.19. Longer extracts of Inform 6 code

Whole routines, object and class definitions (or any other directives) can be pasted in wholesale using sentences like so:

```
Include (-
[ MyInform6Routine a b; return a*b; ];
-).
```

Such inclusions are pasted into the final compiled code at the end of the file, after the I6 grammar has been declared.

In such extracts, we sometimes need to refer to objects, variables or values which can't be described using I6: or rather, which can be described, but we don't know how. To this end, any text in an inclusion written in "(+" and "+)" parentheses is treated as an I7 value, and compiled accordingly, with all type-checking waived for the occasion. For instance:

```
Include (-
Global my_global = (+ the tartan rucksack +);
-).
```

Here "the tartan rucksack" is translated into "O18_tartan_rucksack", or something similar: the I6 object created to represent the rucksack. Thus the actual line of code produced is

```
Global my_global = O18_tartan_rucksack;
```

The material between "(+" and "+)" is generally treated as a value, and thus compiles to the I6 form of that value. But it could also be a property name, which compiles to the I6 form in question, or a defined adjective, which compiles to the name of the

routine to call which tests whether that adjective is true.

Three warnings. The material in "(-" and "-)" is called template code, and it is not quite treated as literal. That means certain characters cause Inform to react:

1. Beware of accidental "(+" usage - for instance,

```
Include (-
[ MyCleverLoop i; for (++i; i<10; i++) print i; ];
-).
```

looks reasonable, but contains "(+" and "+)". Spaces around the first "++" would have been enough to avoid this one; "+)" is only significant where it follows a "(+".

2. Beware of placing an "@" character in the first column, that is, immediately following a new line. (In template code this marks off paragraph divisions.) So for instance,

```
Include (-
[ Set_Stream ret;
@glk 67 ret;
];
-).
```

is tripped up by the Glulx assembly language opcode "@glk" because this occurs in column 1. Indenting it with a little space or a tab is enough to avoid the problem.

3. Be careful if you're creating an I6 variable holding initialised I7 text. For example,

```
Include (-
Global saved_optional_prompt = (+ "!!>" +);
-) after "Definitions.i6t".
```

looks as if it will work, but doesn't, for reference-counting reasons we needn't go into; instead you need

```
Include (-
Array sop_storage --> PACKED_TEXT_STORAGE "!!>";
Global saved_optional_prompt = sop_storage;
-) after "Definitions.i6t".
```

But it's far better to avoid initialising text variables from I6 entirely. The same

problems arise with constant lists.

Start of Chapter 27: Extensions

Back to §27.18. Making and testing use options

Onward to §27.20. Primitive Inform 6 declarations of rules

Example 452: **Status line with centered text, the hard way** A status line which has only the name of the location, centered.

§27.20. Primitive Inform 6 declarations of rules

By writing a sentence like this:

```
The underground rule translates into I6 as "UNDERGROUND_R".
```

we create a new rule, the "underground rule", and also notify Inform that it will have no definition as I7 source text: instead, it will be provided as an I6 routine called "UNDERGROUND_R". We can define this with an Include like so:

```
Include (-
[ UNDERGROUND_R;
    if (real_location hasnt light) { RulebookSucceeds(); rtrue; }
    rfalse;
];
-).
```

The rule should return false if it wants to make no decision, but call either RulebookSucceeds or RulebookFails and return true if it does. These routines can optionally take an argument: which will be the return value from the rulebook.

Note that UNDERGROUND_R itself has no arguments. In the case of an action based rulebook, the I6 variables noun, second and actor can be referred to, while for a value based rulebook the parameter is stored in the I6 global variable parameter_object (which is not necessarily an object, in spite of the name).

We can put this rule into a rulebook in the same way that any named rule can be:

The underground rule is listed in the spot danger rules.

§27.21. Inform 6 objects and classes

As might be expected, I7 compiles an I6 class for each kind, and an I6 object for each of its own objects. We can meddle with its compilation process here using a further refinement of Include. For instance, suppose we want the I6 class definition for things to come out containing a property like this:

```
Class K2_thing ...
    with marmalade_jar_size 6,
    ...
```

How to arrange this? One way is to create an ordinary I7 property, like so:

```
A thing has a number called marmalade jar size. The marmalade jar
size of a thing is usually 6. The marmalade jar size property
translates into I6 as "marmalade_jar_size".
```

(Without that last sentence, the property won't get any familiar name.) But sometimes we need more, and want to actually write new material to go into the definition. This can be done like so:

```
Include (- with before [; Go: return 1; ], -) when defining a vehicle.
```

This glues in a new property to the class compiled to represent the I7 kind "vehicle". (See the DM4 for why. However, since the entire actions machinery is different in the I7 world, note that "after", "react_before" and "react_after" no longer have any effect, and nor does "before" for rooms.)

And similarly:

```
Include (- has my_funny_attribute, -) when defining the hot air
balloon.
```

If we need a particular I7 object or kind to end up with a particular I6 name, we can write:

```
The whatsit object translates into I6 as "whatsit".
The thingummy kind translates into I6 as "thingummy_class".
```

Start of Chapter 27: Extensions
Back to §27.20. Primitive Inform 6 declarations of rules
Onward to §27.22. Inform 6 variables, properties, actions, and attributes

§27.22. Inform 6 variables, properties, actions, and attributes

I7's variables are usually compiled as entries in an array rather than as I6 variables. However, we can instead tell Inform to use an existing I6 variable (either one that we declare ourselves, or one in the I6 template layer). For example:

```
Room description style is a kind of value. The room description styles
are Brief, Verbose and Superbrief.
The current room description style is a room description style that
varies.
The current room description style variable translates into I6 as
"lookmode".
```

This is a feature provided to help I7 source text to use variables internal to the I6 template code. It can, if really necessary, also be used to give I7 names to entirely new I6-level variables, created like so:

```
Include (- Global my_variable = 0; -) after "Definitions.i6t".
```

This style of hybrid coding is really not encouraged.

I7's properties are compiled sometimes as I6 properties, sometimes as I6 attributes, sometimes as bits in a bitmap somewhere. However, we can override I7 by telling it that one of its property names is equivalent to an already-existing I6 property or attribute: if so then I7 will use that name and will not compile any directive to create it. For example:

```
The switched on property translates into I6 as "on".
The initial appearance property translates into I6 as "initial".
```

We do not need to translate "switched off", the opposite to "switched on": I7 will now compile this to "~on".

Lastly, actions can also be translated (though it's usually better to translate their rules instead and invent new I7 actions covering them):

```
The unlocking it with action translates into I6 as "Unlock".
```

Start of Chapter 27: Extensions

Back to §27.21. Inform 6 objects and classes

Onward to §27.23. Inform 6 Understand tokens

§27.23. Inform 6 Understand tokens

The parser which deciphers the player's typed commands is written in I6, and many of the basic tokens of Understand grammar are implemented as "general parsing routines" (GPRs), the specification of which is described fully in the Inform 6 Designer's Manual. I7 translates much of the source text's Understand grammar into GPRs, and once again we can bypass this process and supply an Understand token directly as an I6 GPR. For example:

```
The Understand token squiggle translates into I6 as
"SQUIGGLE_TOKEN".
```

We then have to include a routine of that name into I7's output using the "Include" instruction, on which more later.

This creates a token "[squiggle]"; so for instance if the source text contains:

```
Understand "copy [squiggle]" as ...
```

then Inform would parse the command COPY FIGURE EIGHT by calling the SQUIGGLE_TOKEN routine as a GPR with the word marker at 2, that is, at the word FIGURE.

As always, this should be done only where there seems no better way, or where speed is very important. For any fairly simple range of possibilities, it's better to use the techniques in the Understand chapter, or to use unit specifications.

Start of Chapter 27: Extensions

Back to §27.22. Inform 6 variables, properties, actions, and attributes

Onward to §27.24. Inform 6 adjectives

§27.24. Inform 6 adjectives

There are three ways to specify that an adjective is defined at the I6 level. For example:

```
Definition: a number is prime rather than composite if I6 routine
    "PRIMALITY_TEST" says so (it is greater than 1 and is divisible
only by itself and 1).
```

Inform now actually tests if a number N is prime by calling PRIMALITY_TEST(N), and it assumes that we have also included such a routine in the output. The routine is expected to return true or false accordingly.

The text in brackets does nothing functional, but is the text used in the Lexicon dictionary part of the Phrasebook index for the user's benefit; it should be a brief definition. Extension authors are asked to provide these little definitions, so that their users won't be confused by blank lexicon entries.

The second way makes a more capable adjective, since it can not only be tested, but also made true or false using "now". For example:

```
Definition: a scene is crucial if I6 routine "SceneCrucial" makes it so
    (it is essential to winning).
```

The difference here is "makes it so", not "says so", and as this implies, the routine has more power. "SceneCrucial" is called with two arguments: SceneCrucial(S, -1) tests whether the scene is crucial or not and returns true or false; SceneCrucial(S, true) must make it true; and SceneCrucial(S, false) must make it false. Another useful difference is that if the kind of value is one which is stored in block form (e.g. for an adjective applying to text), the routine is given a pointer to the block, not a fresh copy.

A third way to define an adjective, which should be used only if speed is exceptionally important, is to provide a "schema" - a sort of I6 macro, like those provided by the C preprocessor. For example:

```
Definition: a rulebook is exciting if I6 condition
    "excitement_array-->(*1)==1" says so (it is really wild).
```

The escape "*1" is expanded to the value on which the adjective is being tested. (This is usually faster than calling a routine, but in case of side-effects, the "*1" should occur only once in the condition, just as with a C macro.) To repeat: if in doubt, use the I6 routine method above.

Start of Chapter 27: Extensions

Back to §27.23. Inform 6 Understand tokens

Onward to §27.25. Naming Unicode characters

§27.25. Naming Unicode characters

Inform allows the Unicode characters to be identified either with a decimal number or by name, but it has none of the character names built-in, and for efficiency reasons it only learns them when necessary.

Users normally teach these names to Inform by including one of the extensions "Unicode Character Names" or "Unicode Full Character Names", which consist of many hundreds of sentences like so:

```
anticlockwise open circle arrow translates into Unicode as 8634.
```

Nothing restricts this usage to those extensions.

§27.26. The template layer

When Go is clicked, Inform translates the I7 source text into I6 code, but the directly translated code could not survive on its own: it needs a large body of supporting code, also written in I6, to sustain it. (Just as a program like iTunes or Firefox cannot run on a bare machine, but needs an operating system already up and running to support it.) Until 2008, this supporting code was provided by the I6 library, that is, the standard distribution of useful I6 code supplied with distributions of Inform 6.

However, the supporting code is now generated from a collection of about 35 "segments" of I6 code which together make up "the template layer". The reason for the term "template" is that the segments are not quite directly copied into I7's output, but instead act as templates from which I7 generates code - the final output contains variations according to what the original source text needs.

Each segment has its own name, which looks like a leafname plus the ".i6t" filename extension (which stands for "I6 template"). Internally, a segment is itself divided up into named parts. For instance, the segment "Relations.i6t" contains a part called "Symmetric One To One Relations" which provides I6 routines for changing and testing such relations. There are more than 600 named parts across the template as a whole; it is quite a large program. An annotated, typeset version of the template - amounting to a roughly 500-page book - is available for download from the Inform website.

The most powerful use of "Include" allows code to included before, instead of or after any named part or segment in the template. For example:

```
Include (- ... -) before "Relations.i6t".
Include (- ... -) instead of "Relations.i6t".
Include (- ... -) after "Symmetric One To One Relations" in
"Relations.i6t".
```

Multiple such inclusions can be made for the same segment or part. If so, all will take effect in the case of "before" or "after", but for "instead of" only the most recent one takes effect. Inclusions requested before, or after, a segment or part which has been replaced with "instead of" will take effect and appear before or after the code which appears instead of it.

The pre-2008 syntax

```
Include (- ... -) before the library.
```

has been withdrawn; the new syntax

```
Include (- ... -) after "Definitions.i6t".
```

should have the same effect.

Template files are not written in literal I6, but in a marked-up, annotated form of I6 which has special transcription commands embedded into it. These commands should absolutely not be used except in the built-in template files, with one exception:

```
{-segment:Flowers.i6t}
```

places the whole of the template file "Flowers.i6t" in this position. The built-in template does not of course contain "Flowers.i6t", but Inform allows the optional "I6T" subfolder of the ".materials" folder of a project to hold additional or replacement template files. Thus the project "Botanic Gardens.inform" might store:

```
Botanic Gardens.materials/I6T/Flowers.i6t
```

It could even contain:

```
Botanic Gardens.materials/I6T/Relations.i6t
```

in which case this would automatically be used instead of the built-in copy of "Relations.i6t", without any change to the I7 source text being needed. In this way, projects can (if they need to) use partly or entirely customised templates.

One application of this might allow for chunks of I6 code generated by external utilities - Perl scripts, lex and yacc, or other code generators - by compiling those to suitable template files in .materials/I6T and then using an inclusion like

```
Include (- {-segment:MyStuff.i6t} -).
```

in the I7 source text.

Template hacking, as it's called, is a last resort. If there is any way to achieve the same ends by writing ordinary I7 source text, then that will always be better. If it is possible to write "Include (- ... -)" without mentioning any segment or part, that's much to be preferred, because it has more chance of continuing to work into the future when the template might have been rewritten.

Start of Chapter 27: Extensions

Back to §27.25. Naming Unicode characters

Onward to §27.27. Translating the language of play

§27.27. Translating the language of play

The "language of play" is the natural language used to communicate with the player at run-time: this is normally English.

That means that it is difficult to write, say, Spanish-language IF using Inform 7, though heroic work by the Spanish IF community has overcome this. Inform 6 provided for translation by isolating its linguistic code in a part of the I6 library called the "language definition file", which was normally "English.h". Translations were gradually made to most major European languages, resulting in alternative language definition files called "French.h", "Italian.h" and so on. Full details on how to write a language definition file were given in the Translations chapter of the DM4, that is, the fourth edition of the Inform 6 Designer's Manual.

In I7 the system is different. We use the template, not a library. Instead of providing a language definition file such as "French.h", a translator should create an extension called something like "French Language by Jacques Mensonge". (The language should be named in English, so "French Language by ...", not "Langue français by ...") This extension should then contain broadly the same material as an I6 language definition file, but written in a mostly higher-level way. See the extension "English Language by Graham Nelson" supplied with I7, which is included automatically by default.

Start of Chapter 27: Extensions

Back to §27.26. The template layer

Onward to §27.28. Segmented substitutions

§27.28. Segmented substitutions

A "segmented" substitution is a syntax where text is placed between two or more different text substitutions. Examples include:

```
"This hotel is [if the player is female]just awful[otherwise]basic[end
if]."
"Annie [one of]dances[or]sulks[or]hangs out at Remo's[at random]."
```

To create such syntaxes, it is not enough just to define how each expands into I6 code: for one thing we may need to know about the later terms in order to expand the earlier ones, which is normally impossible, and for another thing, the individual text substitutions mean nothing in isolation. For instance, Inform produces a problem if the following is tried:

```
"The hotel [at random] is on fire."
```

because "[at random]" is only legal when closing a "[one of] ..." construction. But if "[at random]" had been defined as just another text substitution, Inform would not have been able to detect such problems.

Inform therefore allows us to mark text substitutions as being any of three special kinds: beginning, in the middle of, or ending a segmented substitution. There can be any number of alternative forms for each of these three variants. The syntax policed is that

```
(a) Any usage must lie entirely within a single say or piece of text.
(b) It must begin with exactly one of the substitutions marked as
"beginning".
(c) It can contain any number, including none, of the substitutions
marked as "continuing" (if there are any).
(d) It must end with exactly one of the substitutions marked as
"ending".
```

A simple example:

```
To say emphasis on -- beginning say_emphasis_on: (- style
underline; -).
To say emphasis off -- ending say_emphasis_on: (- style roman; -).
```

This creates "[emphasis on]" and "[emphasis off]" such that they can only be used as a pair. The keyword "say_emphasis_on", which must be a valid I6 identifier (and hence a single word), is never seen by the user: it is simply an ID token so that Inform can identify the construction to which these belong. (We recommend that anybody creating such constructions should choose an ID token which consists of the construction's name but with underscores in place of spaces: this means that the namespace for ID tokens will only clash if the primary definitions would have clashed in any case.)

Start of Chapter 27: Extensions

Back to §27.27. Translating the language of play

Onward to §27.29. Invocation labels, counters and storage

Example 453: ★ **Chanel Version 1** Making paired italic and boldface tags like those used by HTML for web pages.

§27.29. Invocation labels, counters and storage

The process of expanding the I6 code which represents a phrase is called "invocation". As we have seen, when a phrase is defined using a single piece of I6 code, invocation consists of copying out that I6 code, except that tokens in braces "{thus}" are replaced:

```
To say (something - number): (- print {something}; -).
```

Ordinarily the only token names allowed are those matching up with names in the prototype, as here, but we have already seen one special syntax: "{phrase options}", which expands as a bitmap of the options chosen. And in fact the invocation language is larger still, as a skim through the Standard Rules will show. The notes below deliberately cover only some of its features: those which are likely to remain part of the permanent design of Inform, and which are adaptable to many uses. **Please do not use any of the undocumented invocation syntaxes: they change frequently, without notice or even mention in the change log.**

The first special syntaxes are textual tricks. {-delete} deletes the most recent character in the I6 expansion of the phrase so far. {-erase} erases the I6 expansion of the phrase so far. {-open-brace} and {-close-brace} produce literal "{" and "}" characters.

The following:

```
{-counter:NAME}
{-counter-up:NAME}
{-zero-counter:NAME}
{-counter-makes-array:NAME}
```

create (if one does not already exist) a counter called NAME. This is initially zero, and can be reset back to zero using "{-zero-counter:NAME}", which expands into no text. The token "{-counter:NAME}" expands into the current value of the counter, as a literal decimal number. The token "{-counter-up:NAME}" does the same, but then also increases it by one. Finally, the token "{-counter-makes-array:NAME}" expands to nothing, but tells Inform to create an "-->" array called "I7_ST_NAME" which includes entries from 0 up to the final value of the NAME counter.

This allows each instance in the source text of a given phrase to have both (i) a unique ID number for that invocation, and (ii) its own word of run-time storage, which can allow it to have a state preserved in between times when it is executed. For example:

```
To say once only -- beginning say_once_only:
    (- {-counter-makes-array:say_once_only}if
(I7_ST_say_once_only-->{-counter:say_once_only} == false) {-open-
brace} I7_ST_say_once_only-->{-counter-up:say_once_only} = true;
-).
To say end once only -- ending say_once_only:
    (- {-close-brace} -).
```

To complete the tools available for defining a segmented substitution, we need a way for the definition of the head to know about the middle segments and the tail:

When invoking either the head or the tail, {-segment-count} expands to the literal decimal number of pieces of text in between the two, which is always one more than the number of middle segments, since the text comes in between the segments. When invoking any middle segment, {-segment-count} expands to the number of pieces of text so far -- thus it expands to 1 on the first middle segment invoked, 2 on the next, and so on.

Lastly {-final-segment-marker} expands to the I6 identifier which marks the end segment, or to I6_NULL if the end segment has no marker. The idea of markers is to enable the head's definition to know which of a number of choices has been used for the tail, supposing that this is a construction with a variety of legal endings. For example:

```
To say emphasise -- beginning say_emphasise:
    (- style {-final-segment-marker}; -).
To say with italics -- ending say_emphasise with marker underline:
    (- style roman; -).
To say with fixed space type -- ending say_emphasise with marker
fixed:
    (- style roman; -).
```

The markers used for the tails here are "underline" and "fixed", and when the head is invoked, the marker for its tail is expanded into the argument of I6's "style" statement.

The examples above are all to do with segmented substitutions, which is where they are most useful, but most of the syntaxes above work equally well for ordinary "To..." phrase definitions.

Start of Chapter 27: Extensions

Back to §27.28. Segmented substitutions

Onward to §27.30. To say one of

§27.30. To say one of

Many of the invocation syntaxes described in the previous section are used in the definition by the Standard Rules of the "[one of] ... [or] ... [purely at random]" construction, so it makes a good example of how they can be used.

First, this is a segmented substitution with a single possible beginning ("[one of]"), a single possible middle ("[or]") but a choice of many possible endings. Almost everything is compiled by the invocation of the beginning:

```
To say one of -- beginning say_one_of (documented at phs_oneof):
(-
    {-counter-makes-array:say_one_of}
    {-counter-makes-array:say_one_flag}
    if (I7_ST_say_one_flag-->{-counter:say_one_flag} == false) {
        I7_ST_say_one_of-->{-counter:say_one_of} = {-final-segment-
marker}(I7_ST_say_one_of-->{-counter:say_one_of},
{-segment-count});
        I7_ST_say_one_flag-->{-counter:say_one_flag} = true;
    }
    if (say__comp == false) I7_ST_say_one_flag-->{-
counter:say_one_flag}{-counter-up:say_one_flag} =
false;
    switch ((I7_ST_say_one_of-->{-counter:say_one_of}{-counter-
```

```
up:say_one_of})%({-segment-count}+1)-1)
{-open-brace}
        0: -).
To say or -- continuing say_one_of (documented at phs_or):
    (- @nop; {-segment-count}: -).
To say purely at random -- ending say_one_of with marker
I7_SOO_PAR (documented at phs_purelyrandom):
    (- {-close-brace} -).
```

The 3rd invocation of this (say) might compile the following:

```
I7_ST_say_one_of-->2 = I7_SOO_PAR(I7_ST_say_one_of-->2, 4);
switch((I7_ST_say_one_of-->2)%5 - 1) {
    0: ... first text ...
    1: ... second text ...
    2: ... third text ...
    3: ... fourth text ...
}
```

First, we notified Inform that it needs to allocate an array (I7_ST_say_one_of) providing storage associated with the counter "say_one_of". This we used to count off individual invocations of "[one of]", so that each would have its own word of storage - for the 3rd invocation, I7_ST_say_one_of-->2. We then call a state-changing routine, in this case I7_SOO_PAR, which is allowed to know the previous state and also the number of options available, and which returns the new state. The state is supposed to be the option chosen last time, but that means that there are not 4, but 5 possibilities: 0 for "there was no last time", then 1 to 4 for the possible outcomes. We reduce the state mod 5 to obtain the decision this time, and subtract 1 because it happens to be convenient to make the switch statement run from 0 to 3 rather than 1 to 4. (The reason we reduce the state mod 5 is to allow the state-changer to squirrel away secret information in the upper bits of the state, if it wants to. Note that subtracting one means that the switch value might be -1, which results in no text being printed: thus if the state-changer chooses 0, it can decide on none of the above.)

In this design, the marker attached to the choice of ending substitution is the name of the I6 state-changer: here is the I7_SOO_PAR routine.

```
[ I7_SOO_PAR oldval count; if (count <= 1) return count; return
random(count); ];
```

As it happens, this ignores the old value: after all, it is meant to be purely at random, and nothing could be less pure than taking the last outcome into consideration when choosing the next.

Note that the counter say_one_of is advanced in invocation of the head. It might seem that the tidier design, somehow, would be to advance the counter in the invocation of the tails, but this is not a good idea. In general it is not safe to assume that the counter will have the same value when the tail is invoked that it had when the head was invoked, because segmented say constructions can legally be nested in Inform strings. Because of this, it is best to deal with a counter entirely in a single invocation, either of the beginning or the ending.

Because "[one of] ... [or] ..." is such a useful construction - switching between alternative forms of text, which writers of IF very often do - the above implementation is intentionally left open for new endings to be added, and the examples below show how easily this can be done.

Start of Chapter 27: Extensions

Back to §27.29. Invocation labels, counters and storage

Example 454: ★ **Blink** Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

Example 455: ★★ **Uncommon Ground** Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

Examples from Chapter 27: Extensions

Start of this chapter

Indexes of the examples

Example **Modern Conveniences**

Suppose we want to write an extension or other portable code which defines a "kitchen" kind of room and a "bathroom" kind of room. All kitchen rooms we create in the future will automatically contain standard kitchen appliances: fridge, freezer, sink with taps, counters, cabinets, and a stovetop with built-in oven. Similarly, all bathrooms will have sinks, baths, cabinets, and toilets, and respond to some standard interactions.

We would do this with a standard assembly:

```
"Modern Conveniences"

Section 1 - Kitchens

A kitchen is a kind of room.

A refrigerator is a kind of container. A refrigerator is
usually closed and openable. A refrigerator is usually
fixed in place. A refrigerator is usually scenery.
Understand "fridge" as a refrigerator.

A freezer compartment is a kind of container. A freezer
compartment is usually closed and openable. A freezer
compartment is part of every refrigerator.
```

Now: we're going to want many of the items in our kitchen to have switches, and to handle input sensibly regardless of whether the player types TURN ON STOVE or TURN ON STOVE SWITCH. (This is apparently a stove with only one burner.) For convenience, we'll define an "includes" relation:

```
Inclusion relates a thing (called X) to a thing (called Y)
```

```
when Y is part of X. The verb to include means the
inclusion relation.
```

```
A stove is a kind of supporter. It is usually scenery.
An oven is a kind of container. An oven is usually
openable and closed. One oven is a part of every stove.
A switch is a kind of device. A switch is part of every
stove. A switch is part of every oven.
Understand "[something related by reversed
incorporation] switch" as a switch.
```

What follows is fairly straightforward, but notice that we are somewhat obsessively naming every rule. This is much more important in extensions (where someone else may need to manipulate our code from within their own source) than it is when we are simply composing source text for ourselves.

```
Setting action variables for opening a stove (this is the
stove-opening rule):
    let relevant oven be a random oven which is part of
the noun;
    now the noun is the relevant oven.
```

```
Setting action variables for switching on something
which includes a switch (this is the redirecting switches
for switching on rule):
    let relevant switch be a random switch which is part of
the noun;
    now the noun is the relevant switch.
```

```
Setting action variables for switching off something
which includes a switch (this is the redirecting switches
for switching off rule):
    let relevant switch be a random switch which is part of
the noun;
    now the noun is the relevant switch.
```

```
Before printing the name of a switch (called target) (this
is the switch identification rule):
```

```
    say "[random thing which includes the target] ".


A sink is a kind of container. A sink is usually fixed in
place and scenery. A tap is a kind of switch. A tap is
part of every sink. Understand "faucet" or "taps" as a
tap. Understand "[something related by reversed
incorporation] tap/faucet/taps" as a tap.


Instead of opening a tap, try switching on the noun.
Instead of closing a tap, try switching off the noun.


Report switching on a tap (this is the standard report
switching taps on rule):
    say "You turn on [the noun]." instead. [since "switch
on" sounds weird in this context.]


Report switching off a tap (this is the standard report
switching taps off rule):
    say "You turn off [the noun]." instead.


After examining something which includes a switched
on tap (called relevant tap) (this is the report flowing
water rule):
    say "The water is flowing from [the relevant tap]."


A drain is a kind of container. A drain is part of every
sink. Understand "plughole" as the drain. Understand
"[something related by reversed incorporation] drain/
plughole" as a drain.


Instead of inserting something into a drain (this is the no
clogging drains rule), say "Pointless."
```

This is probably about as far as we want to go in a generic simulation: it

is tempting to code up water, drains down which the player can lose objects, sinks that get clogged and overflow, and so on; but the more we embellish in these ways, the more likely the end result would be disruptive to individual games. For right now what we're aiming for is something simple which provides the basic interactions a player might expect in this kind of room, but which does not have any significant implications for the surrounding world model.

A particularly conservative author might even want to make it turn out that the water has been shut off and nothing flows from the taps: in the extension documentation, we might want to include a line or two of example showing how this might be done by changing or removing the relevant rules of our extension.

```
A cabinet is a kind of container. A cabinet is usually
openable and closed. It is scenery.
    Understand "cupboard" or "cupboards" or "cabinets"
as a cabinet.


A counter is a kind of supporter. It is scenery.
    Understand "countertop" as a counter.


A cabinet is in every kitchen.
A counter is in every kitchen.
A refrigerator is in every kitchen.
A sink is in every kitchen.
A stove is in every kitchen.


Section 2 - Bathrooms


A bathroom is a kind of room.


A toilet is a kind of supporter. Understand "john" as a
toilet. A toilet is usually fixed in place and enterable and
scenery.
```

```
A bath is a kind of container. A bath is usually a fixed in
place and enterable and scenery. A tap is part of every
bath. A drain is part of every bath. Understand "bathtub"
or "shower" as a bath.
```

```
A sink is in every bathroom.
A toilet is in every bathroom.
A bath is in every bathroom.
A cabinet is in every bathroom.
```

If we were feeling particularly ambitious and inclined toward interior decoration, we could add bath mats, mirrors, plungers, toilet brushes, overhead lighting, towel racks, scented candles, boxes of facial tissue, shampoo bottles, scrubbing loofahs, etc. ad nauseam; but we'll keep it relatively simple for now. Of course, if we have a toilet, we pretty much have to accept that the player will try to make use of it:

```
Understand "flush [toilet]" or "use [toilet]" as a mistake
("You have no need at the moment.").
```

```
Understand "take shower" or "take bath" or "bathe" or
"wash" as bathing. Bathing is an action applying to
nothing.
```

```
Check bathing (this is the restrict baths to bathrooms
rule):
    if the location is not a bathroom, say "This isn't the
place." instead.
```

```
Check bathing (this is the block bathing rule):
    say "You haven't time for a bath." instead.
```

```
Washing is an action applying to one thing. Understand
"clean [something]" or "wet [something]" or "wash
```

```
[something]" as washing.


Instead of washing the player, try bathing.


Check washing (this is the restrict washing to the
proximity of sinks rule):
    unless the player can touch a sink, say "This isn't the
place." instead.


Check washing (this is the block washing rule):
    say "It doesn't seem worth the bother." instead.
```

Now we might put this to work in a short example.

One slight challenge lies in giving these assembled pieces separate descriptions. When we have an assembly that adds parts to objects, we can then talk about (for instance) "the stove's switch" elsewhere in the code. But items that have been assigned rooms are not named in the same way, so we cannot talk about "the Industrial Kitchen's stove" in our code as a way to assign it a description or special behavior. In quite a simple example, we could make the descriptions of the kind simply be the descriptions we want for the individual items:

```
Section 3 - An Example We Might Offer


Our Household Kitchen is a kitchen.


The Tiny Bathroom is a bathroom. It is west of Our
Household Kitchen.


The description of a stove is "Scrupulously polished."
```

```
The description of a refrigerator is "It is baby blue and
has the contours of a 50[']s chevy. One of these days it
really will break down, but it's been serving your family
faithfully since your grandmother's honeymoon."
```

```
Test me with "x refrigerator / open fridge / x freezer /
look in freezer / open freezer / turn on stove / turn on
oven / x oven switch / turn off oven switch / turn off
stove switch / turn on taps / x sink / w / x sink / turn on
sink / take bath / use toilet".
```

In a game that featured multiple bathrooms and kitchens, this wouldn't be enough; our author might give the stove kind (say) a description that checked its location, as in

```
The description of a stove is "[if in Industrial Kitchen]A
massive stainless steel stove-top with six
burners[otherwise]Your standard four-burner item[end
if]."
```

or create an

```
Instead of examining the stove in the Industrial Kitchen:
...
```

sort of rule for those objects he wanted to describe specially; or he might use a when play begins rule to initialize a few things:

```
When play begins:
    let N be a random stove in the Industrial Kitchen;
    move the boiling pot to N;
    change the description of N to...
```

Or we might even (if we anticipate lots of these kinds of amendments) want to rig up something more complex that finds the descriptions of appliances in a table, rather than relying on their individual description properties. This can all be done, but it is only interesting as long as it remains genuinely labor-saving: that is, as long as the convenience of having the assembly is greater than the annoyance of writing special rules to cover for the automation. In the end, the "kitchen" and "bathroom" room types are likely to be most useful for authors who want

to include the standard props but not actually make them a critical part of the game; if stoves and sinks have more of a starring role in the production, authors may be better off coding them or at the very least placing them by hand, as in

```
The Industrial Kitchen is a room.

Thor is a stove in the Industrial Kitchen. It supports a
boiling pot.
```

All these quirks are things that we (as the extension author) want to think out in advance: we should ideally warn authors about possible pitfalls in using our extension (if we can think of them) and point out ways of customizing the behavior (if there are interesting ways).

---

450 **ExampleTilt 3** *RB*

Displaying the card suits from our deck of cards with red and black colored unicode symbols.

Books and articles about card-playing traditionally abbreviate card names into a simple two-symbol notation: a number or letter representing the card rank, followed by a symbol indicating the card suit. Suppose that we want to emulate this notation when taking inventory in our poker game.

The trick here is that colored output is done in different ways by the Z-Machine and by Glulx, so we'll need two different versions of the same section in order to produce this output. The relevant source is right at the beginning:

```
"Tilt"

Section 0 - Colored Output in Two Forms
```

For the suit symbols, we'll want the Unicode extension included with Inform:

```
Include Unicode Character Names by Graham Nelson.


Rule for printing the name of a card (called target) while
grouping together:
    say "[rank of the target as abbreviated value][suit of
the target as symbol]".


To say (current suit - a suit) as symbol:
    if current suit is diamonds, say "[red letters][unicode
black diamond suit][default letters]";
    if current suit is spades, say "[unicode black spade
suit]";
    if current suit is clubs, say "[unicode black club suit]";
    if current suit is hearts, say "[red letters][unicode
black heart suit][default letters]".


Section 0Z (for Z-machine only)
```

The Basic Screen Effects extension bundled with Inform includes mechanisms to change the text color, so for the Z-machine, we need only include this:

```
Include Basic Screen Effects by Emily Short.


Section 0G (for Glulx only)
```

Under Glulx, we need slightly more set-up: Glulx requires that we define special user font styles when we plan to make display changes. A fuller discussion of this (and of how to define new colors) appears in the documentation of "Glulx Text Effects", but an implementation sufficient to our purposes would be

```
Include Glulx Text Effects by Emily Short.
```

```
Table of User Styles (continued)

style name
special-style-1
```

```
Table of Common Color Values (continued)

assigned value
red
```

```
To say red letters: say first custom style.
```

```
To say default letters: say roman type.
```

From here, the rest of the source is mostly as we've seen in previous examples:

```
Section 1 - Cards
```

```
Suit is a kind of value. The suits are hearts, clubs, diamonds, and spades. Understand "heart" as hearts. Understand "club" as clubs. Understand "diamond" as diamonds. Understand "spade" as spades.
```

```
A card is a kind of thing. A card has a suit. A card has a number called rank. Understand the suit property as describing a card. Understand the rank property as describing a card.
```

52 cards are in the card repository.

To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".

Rule for printing the name of a card (called target):
    say "[rank of the target as a card value] of [suit of the target]"

To say (count - a number) as abbreviated value:
    choose row count in the Table of Value Names;
    say "[abbrev entry]".

Table of Value Names

term	abbrev
"ace"	"A"
"deuce"	"two"
"three"	"3"
"four"	"4"
"five"	"5"
"six"	"6"
"seven"	"7"
"eight"	"8"
"nine"	"9"
"ten"	"10"
"jack"	"knave/J"
"queen"	"Q"
"king"	"K"

After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
            cut the matched text;
    repeat through the Table of Value Names:
        while the player's command includes topic entry:
            replace the matched text with value entry.

```
When play begins:
    reconstitute deck.


To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;
            now item is in the deck of cards;
        now current suit is the suit after the current suit.
```

Section 2 - The Deck and the Discard Pile

```
The Empty Room is a room. "Nothing to see here."


The deck of cards is in the Empty Room. It is a closed
unopenable container. The description is "A standard
poker deck."


The discard pile is a closed unopenable container. The
description is "Cards in this game are discarded face-
down, so the discard pile is not very interesting to see.
All you can observe is that it currently contains [if the
number of cards which are in the discard pile is less
than ten][the number of cards which are in the discard
pile in words][otherwise]about [the rounded number of
cards which are in the discard pile in words][end if]
card[s]."


To decide what number is the rounded number of
(described set - a description of objects):
```

```
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.
```

Rule for printing room description details of something:
do nothing instead.

Section 3 - Drawing and Discarding Actions

Understand the commands "take" and "carry" and "hold" and "get" and "drop" and "throw" and "discard" as something new.

Understand "take [text]" or "get [text]" or "drop [text]" as a mistake ("Here, you only draw and discard. Nothing else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as drawing. Drawing is an action applying to nothing. The drawing action has an object called the card drawn.

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater

than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

Understand "discard [card]" as discarding. Discarding is an action applying to one thing.

Check discarding:
    if the player does not carry the noun, say "You can only discard cards from your own hand." instead.

Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile to the location.

Report discarding:
    say "You toss [the noun] nonchalantly onto the discard pile."

Section 4 - Assessing Hands

Before listing contents while taking inventory: group cards together.

```
Before grouping together cards:
    if the number of cards carried by the player is 5:
        say "[run paragraph on]";
        follow the hand-ranking rules;
        if the rule succeeded, say "[the outcome of the
rulebook]";
        otherwise say "some random cards";
        if the outcome of the rulebook is pair outcome, say
" of [rank of the first thing held by the player as a card
value]s";
    otherwise:
        say "[number of cards carried by the player in
words] assorted cards";
    say " (".



Rule for grouping together cards:
    say "[list hand]".



To say list hand:
    let chosen card be the first thing held by the player;
    while chosen card is a card:
        say "[chosen card]";
        now chosen card is the next thing held after
chosen card;
        if chosen card is a card, say "-".



After grouping together cards:
    say ")".



The hand-ranking rules is a rulebook. The hand-ranking
rules have outcomes royal flush, straight flush, four of a
kind, full house, flush, straight, three of a kind, two pairs,
pair, high card.



The hand-ranking rulebook has a truth state called the
flushness.
The hand-ranking rulebook has a truth state called the
```

straightness.

The hand-ranking rulebook has a number called the pair count.
The hand-ranking rulebook has a number called the triple count.
The hand-ranking rulebook has a number called the quadruple count.

A card can be sorted or unsorted. A card is usually unsorted.

Definition: a card is high if its rank is 11 or more.
Definition: a card is low if its rank is 4 or less.

A hand-ranking rule (this is the initial sort rule):
    now every card is unsorted;
    while the player carries an unsorted card:
        let item be the lowest unsorted card held by the player;
        move item to the player;
        now the item is sorted;
    if sort-debugging is true, say "-- after initial sort: [list hand]".

A hand-ranking rule (this is the finding flushness rule):
    let called suit be the suit of a random card carried by the player;
    if every card carried by the player is called suit, now flushness is true.

A hand-ranking rule (this is the finding straightness rule):
    now straightness is true;
    let N be the rank of the highest card which is carried by the player;

```
    repeat with current rank running from N - 4 to N:
        now the test rank is the current rank;
        unless the player carries a matching card:
            if the current rank is N - 4 and the current rank is
9 and the player carries an ace card, do nothing;
            otherwise now straightness is false.
```

A card can be quadrupled, tripled, paired or uncombined.

Test rank is a number that varies. Definition: a card is matching if its rank is the test rank.

```
A hand-ranking rule (this is the counting multiples rule):
    now every card is uncombined;
    repeat with current rank running from 1 to 13:
        now test rank is current rank;
        let N be the number of matching cards held by the
player;
        if N is 4:
            increment the quadruple count;
            now every matching card held by the player is
quadrupled;
        if N is 3:
            increment the triple count;
            now every matching card held by the player is
tripled;
        if N is 2:
            increment the pair count;
            now every matching card held by the player is
paired.
```

```
A hand-ranking rule (this is the move aces up unless
there's a low straight rule):
    unless the straightness is true and the lowest card
carried by the player is an ace card and the rank of the
highest card carried by the player is 5,
        now every ace card which is carried by the player
is carried by the player;
    if sort-debugging is true, say "-- after ace movement
rule: [list hand]".
```

```
A hand-ranking rule (this is the move pairs forward rule):
    while the player carries a paired card:
        let selection be the lowest paired card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after pairs movement:
[list hand]".


A hand-ranking rule (this is the raise ace pairs rule):
    if the player carries exactly two ace cards:
        repeat with item running through ace cards which
are carried by the player:
            move item to the player;
    if sort-debugging is true, say "-- after paired-ace
movement: [list hand]".


A hand-ranking rule (this is the move multiples forward
rule):
    while the player carries a tripled card:
        let selection be the lowest tripled card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    while the player carries a quadrupled card:
        let selection be the lowest quadrupled card which
is carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after multiples
movement rule: [list hand]".


Definition: a card is ace if its rank is 1.
Definition: a card is king if its rank is 13.


A hand-ranking rule (this is the royal-flush rule):
    if flushness is true and straightness is true and the
highest card carried by the player is king and the lowest
card carried by the player is ace, royal flush.
```

A hand-ranking rule (this is the straight-flushes rule):
    if flushness is true and straightness is true, straight flush.

A hand-ranking rule (this is the four-of-a-kind rule):
    if the quadruple count is 1, four of a kind.

A hand-ranking rule (this is the full-house rule):
    if the pair count is 1 and the triple count is 1, full house.

A hand-ranking rule (this is the flushes rule):
    if flushness is true, flush.

A hand-ranking rule (this is the straights rule):
    if straightness is true, straight.

A hand-ranking rule (this is the three-of-a-kind rule):
    if triple count is 1, three of a kind.

A hand-ranking rule (this is the two-pair rule):
    if the pair count is 2, two pairs.

A hand-ranking rule (this is the pair rule):
    if the pair count is 1, pair.

A hand-ranking rule (this is the default rule):
    high card.

Sort-debugging is a truth state that varies.

Section 5 - Testing hand identification - Not for release

Understand "debug sorting" as debugging hand sorting.
Debugging hand sorting is an action out of world.

Carry out debugging hand sorting:
    if sort-debugging is false, now sort-debugging is true;
    otherwise now sort-debugging is false.

Report debugging hand sorting:
    say "Sort debugging is now [if sort-debugging is true]on[otherwise]off[end if]."

Test me with "draw / g / g / g / g / force hand / g / g / g / g / g / g / g / g / g / g / g".

Table of Testing Hands

set rank	[royal flush]	[straight flush]	[four of a kind]
13	spades	clubs	hearts
12	spades	clubs	diamonds
11	spades	clubs	spades
10	spades	clubs	clubs
9	clubs	clubs	clubs
8	clubs	clubs	clubs
7	hearts	clubs	clubs
6	spades	clubs	clubs

8 clubs
8 clubs
[full house]
1 spades
10 hearts
10 spades
10 clubs
[flush]
5 hearts
7 hearts
11 hearts
12 hearts
[straight]
13 spades
12 diamonds
11 clubs
10 hearts
[three of a kind]
2 spades
2 clubs
4 clubs
3 spades
[two pair]
6 spades
7 clubs
7 diamonds
9 hearts
[two pair, ace high]
6 spades
1 clubs
7 diamonds
1 hearts
[pair]
12 spades
10 diamonds
3 spades
4 clubs
[high]
11 hearts
9 spades
7 clubs
5 diamonds
[tricky sorting: low straight]
2 diamonds
3 spades
4 diamonds
5 diamonds

```
Understand "force hand" as forcing a hand. Forcing a
hand is an action out of world.



Current marker is a number that varies.



Carry out forcing a hand:
    repeat with item running through cards which are
carried by the player:
        increment current marker;
        if current marker is greater than the number of filled
rows in the Table of Testing Hands, now current marker
is 1;
        choose row current marker in the Table of Testing
Hands;
        now the suit of item is the set suit entry;
        now the rank of item is the set rank entry.



Report forcing a hand:
    try taking inventory.
```

---

451 **Example** **Pink or Blue** *RB*

Asking the player to select a gender to begin play.

Suppose we would like to allow the player to choose a gender for the main character. We'd also like this to happen at the beginning of the game and outside the main parsing sequence. "When play begins" seems like a good place to put this.

```
"Pink or Blue"



When play begins:
    say "Should your character be male or female? >";
    if men win, now the player is male;
```

```
    otherwise now the player is female;
    say paragraph break.
```

Now a piece of Inform 6 code handles the unusual input. It's not necessary to understand this to use it, and the code should work for any question you'd like to ask the player. The first three words in quotation marks ('male', 'M', 'man'...) correspond to positive feedback; the later three words correspond to negative feedback. So "to decide whether men win" will be true if the player types one of the first three, and false if he types one of the last three.

```
To decide whether men win:
    (- Question('male','M//','man','female','F//','woman') -)


Include (-


[ Question pos1 pos2 pos3 neg1 neg2 neg3
first_word_typed;
    while (true) {
        VM_ReadKeyboard(buffer, parse);
        wn = 1; first_word_typed = NextWordStopped();
        if (first_word_typed == pos1 or pos2 or pos3) rtrue;
        if (first_word_typed == neg1 or neg2 or neg3)
rfalse;
        print "Please choose ", (address) pos1, " or ",
(address) neg1, ". > ";
    }
];


-)


Instead of examining the player when the player is
female:
    say "Congratulations, you are a girl!"


Instead of examining the player when the player is male:
```

```
    say "Congratulations, you are a boy!"

The Room of Self-Knowledge is a room. "Mirrors cover
every available wall-surface of this hexagonal chamber,
allowing you to examine yourself from all angles."
```

---

452 **Example** **Status line with centered text, the hard way** *RB*

A status line which has only the name of the location, centered.

Making major changes to display features, such as the construction of the status line, sometimes requires that we rely on Inform 6 in bulk; here is how we might produce the Trinity-style status line, with the location centered at the top center of the screen.

```
"Corner of No and Where"

No is a room. Where is west of No.

Rule for constructing the status line:
    print the location in the center of the status line;
    rule succeeds.

To print the location in the center of the status line:
    (- PrintCenteredStatus(); -).

Include (-

Array printed_text --> 64;
```

```
[ PrintCenteredStatus i j;
    @set_cursor 1 0;
    i = 0->33;
    spaces(i);
    @output_stream 3 printed_text;
    print (name) location;
    @output_stream -3;
    j = (i - (printed_text-->0))/2;
    @set_cursor 1 j;
    print (name) location;
    spaces j-1;
];



-)



Test me with "w / e".
```

In fact, as we've already seen, many extra modifications to the display behavior are possible using Basic Screen Effects.

---

453 ExampleChanel Version 1 *RB*

Making paired italic and boldface tags like those used by HTML for web pages.

HTML uses angled brackets to achieve effects, and places italicised text between and tags; and similarly boldface between and . We can mimic this very easily by setting each up as a segmented substitution:

```
"Chanel Version 1"



To say i -- beginning say_i -- running on: (- style
underline; -).
To say /i -- ending say_i -- running on: (- style roman; -).
```

```
To say b -- beginning say_b -- running on: (- style bold;
-).
To say /b -- ending say_b -- running on: (- style roman;
-).



Place Vendôme is a room. "[i]Fashion fades, only style
remains the same[/i] ([b]Coco Chanel[/b]). And this
elegant drawing-room, once a milliner's shop, is a case
in point."



Instead of going nowhere, say "[i]Don't spend time
beating on a wall, hoping to transform it into a door.[/i]
([b]Coco Chanel[/b]) This one is a wall.".



Test me with "look / e".
```

We have had to use square instead of angle brackets, but then, "in order to be irreplaceable one must always be different" (Coco Chanel).

(Marking these as substitutions which run on prevents unexpected paragraph breaks if they should appear immediately after the end of a sentence.)

---

454 **ExampleBlink** *RB*

Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

Suppose we are writing a game in which the mood of the piece changes, and we would like to have lots of descriptions that vary according to its current state. We might in that case want to create our own "by atmosphere" token, to control text variations, like this:

```
"Blink"
```

```
Atmosphere is a kind of value. The atmospheres are
normal, melancholy, and creepy.


The current atmosphere is an atmosphere that varies.


To say by atmosphere -- ending say_one_of with
marker I7_SOO_ATM:
    (- {-close-brace} -).
```

Since we're operating within the untyped Inform 6, we can make use of the fact that kinds of value are (internally) just constants, enumerated in the same order in which they were originally defined. In other words, "normal" at the I6 level translates to 1, "melancholy" to 2, and "creepy" to 3; so we can return the value of the current atmosphere, and thereby select option 1, 2, or 3:

```
Include (-
[ I7_SOO_ATM oldval count;
    if (count < (+ current atmosphere +)) return count;
    return (+ current atmosphere +); ];
-)
```

And that concludes the hard part. Now to test that it works:

```
The Flat is a room. "A small [one of]but
cozy[or]depressing[or]imprisoning[by atmosphere] flat.
Outside the window, the sun is [one of][or][or]apparently
[by atmosphere]shining and there is a brisk breeze
through the leaves of the birch trees. [one of]It would be
quite nice weather for a walk[or]The rest of the world
apparently has no appreciation of what you suffer[or]It
all looks deceptively normal[by atmosphere]."


Instead of waiting when the current atmosphere is
normal:
    say "Everything stretches wide and flat for just a
moment, as though all the world around you were
painted on a thin rubber sheet that is being [italic
type]stretched[roman type]. Then it snaps back into
```

```
place, leaving your ears ringing. But that little glitch was
enough to warn you. Someone is tampering with space-
time again. Someone very close by.";
    now the current atmosphere is creepy.



Test me with "look / z / look".
```

ExampleUncommon Ground *RB*

Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

A slightly more challenging case than the "by atmosphere" example is one in which we want to create text variations depending on the identity of our player character.

What we want to do is build a switch statement in I6, one that looks something like

```
switch(player)
{
    yourself: print "quaint";
    Lolita: print "thrilling";
    default: print "squalid";
}
```

out of I7 that looks like this:

```
say "[show to yourself]quaint[to Lolita]thrilling[to
everyone else]squalid[end show]".



"Uncommon Ground"



The Mud Village is a room. "You stand at the center of a
```

[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show] mud village."

Leforge is a man in the Mud Village. Lolita is a woman in the Mud Village.

Instead of waiting:
    if the player is Lolita, now the player is Leforge;
    if the player is yourself, now the player is Lolita;
    say "You jump bodies. Whoops!"

To say show to (N - a person) -- beginning say_seen_by:
    (-
        switch(player)
        {-open-brace}
        {N}:
    -).

To say to (N - a person) -- continuing say_seen_by:
    (-
        {N}:
    -).

To say to everyone else -- continuing say_seen_by:
    (-
        default:
    -)

To say end show -- ending say_seen_by:
    (-
        {-close-brace}
    -)

Test me with "look / z / look / z / look".

Part II. The Inform Recipe Book

Chapter 1: How to Use The Recipe Book
Chapter 2: Adaptive Prose
Chapter 3: Place
Chapter 4: Time and Plot
Chapter 5: The Viewpoint Character
Chapter 6: Commands
Chapter 7: Other Characters
Chapter 8: Vehicles, Animals and Furnitu
Chapter 9: Props: Food, Clothing, Money
Chapter 10: Physics: Substances, Ropes
Chapter 11: Out Of World Actions and Eff
Chapter 12: Typography, Layout, and Mu
Chapter 13: Testing and Publishing

Start reading here: §1.1. Preface
Part I. Writing with Inform
Indexes of the examples and definitions

Chapter 1: How to Use The Recipe Book

§1.1. Preface; *§1.2. Acknowledgements*; *§1.3. Disenchantment Bay*; *§1.4. Information Only*

Contents of *The Inform Recipe Book*
Chapter 2: Adaptive Prose
Indexes of the examples

§1.1. Preface

The Inform Recipe Book is one of two interlinked books included with Inform 7: a comprehensive collection of examples, showing the practical use of Inform. The other book is **Writing with Inform**, a systematic manual for the software. If you are reading this within the Inform application, you will see that the Recipe Book pages are on "yellow paper", while the manual is on "white paper".

The Recipe Book assumes that the reader already knows the basics covered in Chapters 1 and 2 of *Writing with Inform*: enough to get simple projects working in the Inform application. It's helpful, but not necessary, to have some familiarity with the main ingredients of Inform. For instance, the reader who can play and test the following source text, and who can take a guess at what it ought to do, should be fine:

```
"The Power of the Keys"

Afterlife is a room. "Fluffy white clouds gather round you here in the
afterlife." The Pearly Gates are a door in Afterlife. "The Pearly Gates
- large, white, wrought-iron and splendidly monumental - stand above
you." Heaven is a room. The Gates are above the Afterlife and below
Heaven.

St Peter is a man in the Afterlife. "St Peter, cheery if absent-minded,
studies his celestial clipboard."

Before going through the Pearly Gates:
    say "St Peter coughs disarmingly. 'If you'd read your Bible,' he
says, 'you might recall Revelation 21:21 saying that the twelve gates
were twelve pearls, each gate being made from a single pearl. I really
don't know why people keep imagining it like the entrance to some
sort of public park - oh, well. In you go.'";
    end the story.

Test me with "enter gates".
```

The *Recipe Book* is not a tutorial - it offers advice and examples to crib from, not theory or systematic teaching. The examples here are provided with the express intention that authors cut and paste useful passages into their own works, modifying as they go. This is an excellent way to get things working quickly.

In the traditional saying: good programmers write good code, but great programmers steal it. (Appropriately enough, nobody seems to know who said this first.) For the avoidance of any doubt - the example text is here to be taken, and this infringes no copyright, and requires no acknowledgement. So steal at will. The examples are a part of Inform itself, and as such, they are available to anyone who accepts the Inform licence.

Many programming languages for conventional computing, such as C, come with elaborate libraries of ready-written code - so elaborate, in fact, that they often need much larger manuals than the language itself, and can be hard to learn. Even expert programmers typically use only a small part of what is available in such libraries, giving up on the rest as too complex to use, or too difficult to find out about, or not

quite what they need.

The designers of Inform chose not to go down this road. Rather than providing a general system for liquids (say), which would have to be a quite complicated and opaque program, Inform provides a choice of examples showing how to get different effects. The writer can read the text which achieves these effects, and can simply cut and paste whatever might be useful, and rewrite whatever is not quite wanted.

The wider community of Inform writers has made a great wealth of material available in the form of Extensions, too, and under a Creative Commons Attribution licence requiring only a namecheck: we don't cover the Extensions in this book, because it would grow far too long and be a constant labour to maintain, but it's well worth seeing what is out there.

See **Acknowledgements** *for a chance to try out the cross-referencing links in the Recipe Book - click on the red asterisk or the name of the destination to go there*

Start of Chapter 1: How to Use The Recipe Book
Onward to §1.2. Acknowledgements

Example 1: **About the examples** An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

Example 12: **Midsummer Day** A few sentences laying out a garden together with some things which might be found in it.

---

§1.2. Acknowledgements

David Fisher's "Past raif topics" pages on the Interactive Fiction Wiki were an invaluable tool during the early design of these examples, as they catalog an enormous assortment of implementation problems encountered by IF authors over the past fifteen years.

Thanks also go to Nick Montfort for several conversations during the development of Inform: these inspired a number of ideas about how the author should be able to

control the textual output of a story, and suggested specific problem areas to work on.

Jeff Nyman provided extensive feedback about using *Writing with Inform* in workshops of aspiring IF authors from both programming and conventional fiction writing backgrounds. His observations about the concerns of conventional writers first encountering IF were especially useful, and had a great influence on the organization of the *Recipe Book*. While the results may not meet all the needs he identified, we hope to have taken a step in the right direction.

A few examples were contributed by denizens of rec.arts.int-fiction: Jesse McGrew, Jon Ingold, Mike Tarbert, Eric Rossing, and Kate McKee offered such elegant implementations of various tasks that we have folded their contributions (with permission) into the *Recipe Book*.

Finally, these pages owe much to the questions and suggestions of Inform users on rec.arts.int-fiction and ifMUD.

Start of Chapter 1: How to Use The Recipe Book
Back to §1.1. Preface
Onward to §1.3. Disenchantment Bay

---

§1.3. Disenchantment Bay

"Disenchantment Bay" is a simple work of IF used as a running example in Chapter 3 of *Writing with Inform* - not so much a tutorial as a convenient hook on which to hang some demonstrations of the basics. Because the resulting examples only use basic features and in the most straightforward way, they make for uninteresting "recipes" - so they are not included in the *Recipe Book* proper. But some readers might like to have all twelve stages of the example gathered on a single page: this is that page.

Start of Chapter 1: How to Use The Recipe Book
Back to §1.2. Acknowledgements
Onward to §1.4. Information Only

Example 14: ★ **Disenchantment Bay 1** A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

Example 15: **Disenchantment Bay 2** Disenchantment Bay: creating some of the objects in the cabin's description.

Example 17: **Disenchantment Bay 3** Disenchantment Bay: adding a view of the glacier.

Example 18: **Disenchantment Bay 4** Disenchantment Bay: fleshing out the descriptions of things on the boat.

Example 20: **Disenchantment Bay 5** Disenchantment Bay: adding the door and the deck to our charter boat.

Example 23: **Disenchantment Bay 6** Disenchantment Bay: locking up the charter boat's fishing rods.

Example 25: **Disenchantment Bay 7** Disenchantment Bay: making the radar and instruments switch on and off.

Example 30: **Disenchantment Bay 9** Disenchantment Bay: enter the charter boat's Captain.

Example 33: **Disenchantment Bay 10** Disenchantment Bay: things for the player and the characters to wear and carry.

Example 34: **Disenchantment Bay 11** Disenchantment Bay: making a holdall of the backpack.

Example 28: **Disenchantment Bay 8** Disenchantment Bay: a pushable chest of ice for the boat.

Example 37: ★★★★ **Disenchantment Bay 12** A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

---

§1.4. Information Only

One last preliminary: a handful of the examples do not show how to do anything at all, but are really sidebars of information. Those examples are gathered below, since they contribute nothing by way of recipes.

Start of Chapter 1: How to Use The Recipe Book

Back to §1.3. Disenchantment Bay

Onward to Chapter 2: Adaptive Prose: §2.1. Varying What Is Written

Example 415: ★ **About Inform's regular expression support** Some footnotes on Inform's regular expressions, and how they compare to those of other programming languages.

Example 223: ★★★ **Formal syntax of sentences** A more formal description of the sentence grammar used by Inform for both assertions and conditions.

Example 243: ★★★ **Mathematical view of relations** Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

Example 244: ★★★ **Graph-theory view of relations** Some notes on relations from the point of view of graph theory.

Example 399: ★★★ **Backus-Naur form for rules** The full grammar Inform uses to parse rule definitions, in a standard computer-science notation.

Examples from Chapter 1: How to Use The Recipe Book

Start of this chapter

Chapter 2: Adaptive Prose

Indexes of the examples

★ **Example** **About the examples** *WI*

An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

This is the first of about 400 numbered examples. In a few cases, such as this one, they provide a little background information, but almost all demonstrate Inform source text. The techniques demonstrated tend to be included either because they are frequently asked for, or because they show how to achieve some interesting effect.

The same examples are included in **both** of the books of documentation, but in a different order: in *Writing with Inform*, they appear near the techniques used to make them work; in *The Inform Recipe Book*, they are grouped by the effects they provide. For instance, an example called "Do Pass Go", about the throwing of a pair of dice, appears in the "Randomness" section of *Writing with Inform* and also in the "Dice and Playing Cards" section of *The Inform Recipe Book*. Clicking the italicised WI and RB buttons at the right-hand side of an example's heading switches between its position in each book.

Many computing books quote excerpts from programs, but readers have grown wary of them: they are tiresome to type in, and may only be fragments, or may not ever have been tested. The authors of Inform have

tried to avoid this. All but two dozen examples contain entire source texts. A single click on the paste icon (always placed just left of the double-quoted title) will write the complete source text into the Source panel. All that is then required is to click the Go button, and the example should translate into a working game.

In most cases, typing the single command TEST ME will play through a few moves to show off the effect being demonstrated. (You may find it convenient to create a "scratch" project file for temporary trials like this, clearing all its text and starting again with each new test.)

As part of the testing process which verifies a new build of Inform, each example in turn is extracted from this documentation, translated, played through, and the resulting transcript mechanically checked. So the examples may even work as claimed. But the flesh is weak, and there are bound to be glitches. We would welcome reports, so that future editions can be corrected.

Each example is loosely graded by difficulty: if they were exercises in a textbook, the asterisks would indicate how many marks each question scores. As a general rule:

★ - A simple example, fairly easily guessed.
★★ - A complicated or surprising example.
★★★ - An example needing detailed knowledge of many aspects of the system.
★★★★ - A complete scenario, containing material not necessarily relevant to the topic being demonstrated.

In general, the main text of *Writing with Inform* tries never to assume knowledge of material which has not yet appeared, but the trickier examples almost always need to break this rule.

★ **Example Midsummer Day**
A few sentences laying out a garden together with some things which might be found in it.

"Midsummer Day"

East of the Garden is the Gazebo. Above is the Treehouse. A billiards table is in the Gazebo. On it is a trophy cup. A starting pistol is in the cup. In the Treehouse is a container called a cardboard box.

Test me with "up / x box / d / e / x table / x cup / x pistol / get cup".

---

14 ExampleDisenchantment Bay 1 *WI*

A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

To begin with the title:

"Disenchantment Bay"

There are many Disenchantment Bays across the world, named by eighteenth-century ships' captains - one in Antarctica, another in Tasmania, for instance. The most famous is probably the one where Lewis and Clark's expedition broke through to the Pacific. But ours is the one in Alaska, named in 1791 by a Spanish navigator who had hoped it might lead to the fabled Northwest Passage, and all of this history is beside the point since the game is set in the present day.

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

We might want to start with the glass case.

```
The Cabin contains a glass case. In the glass case is a
collection of fishing rods.
```

Now Inform will have guessed that the case is a container, but its default idea of a container is something like a bucket: permanently open and not able to be opened and shut. We can change that:

```
The case is closed, transparent, and openable.
```

We get a similar set of guesses if we write

```
The bench is in the cabin. On the bench are some blue
vinyl cushions.
```

Using "some" rather than "a" or "the" tells Inform that the cushions are to be referred to as a plural object in the future. And because of the "on the bench..." phrase, Inform will guess that the bench is a supporter and that it is fixed in place and cannot be moved from room to room. We do have to tell it that the bench can be sat on, though:

```
The bench is enterable.
```

And now a short script, so that if we type TEST ME, we experiment with the case and bench:

```
Test me with "examine case / get rods / open case / get
rods / sit on bench / take cushions / get up"
```

If we compile our last version of the cabin, we get a room where the glass case and the bench are listed separately from the room description, even though they have already been mentioned once. We can prevent this by making the already-mentioned things scenery:

```
"Disenchantment Bay"

The Cabin is a room. "The front of the small cabin is
entirely occupied with navigational instruments, a radar
display, and radios for calling back to shore. Along each
side runs a bench with faded blue vinyl cushions, which
can be lifted to reveal the storage space underneath. A
glass case against the wall contains several fishing
rods.

Scratched windows offer a view of the surrounding bay,
and there is a door south to the deck. A sign taped to
one wall announces the menu of tours offered by the
Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a
collection of fishing rods. The case is closed,
transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue
vinyl cushions. The bench is enterable and scenery. The
cushions are scenery.
```

Generally speaking, it is a good idea to recognize the player's attempts to interact with any objects mentioned in the room description, so we should also provide

```
Some navigational instruments, some scratched
windows, a sign, a radar display, and some radios are
scenery in the cabin.
```

```
Test me with "examine instruments / x windows / x sign
/ x display / x radios".
```

The door and the view will need to be done as well, but they are special cases which we will get to shortly.

As noted, making something scenery also means that the player will be prevented from picking it up and carrying it away. This is sensible, though: if an object can be removed from the room where it first appears, we should be careful about mentioning it in the main room description; otherwise, it will continue to be described as present even when someone has carried it off.

---

17 **ExampleDisenchantment Bay 3** *WI*

Disenchantment Bay: adding a view of the glacier.

Suppose we wanted to have the glacier visible from the Cabin of our boat, and anywhere else we might add to the game:

```
The view of the Malaspina glacier is a backdrop. It is
everywhere. The description is "The Malaspina glacier
covers much of the nearby slope, and -- beyond it -- an
area as large as Rhode Island."
```

---

18 **ExampleDisenchantment Bay 4** *WI*

Disenchantment Bay: fleshing out the descriptions of things on the boat.

Currently we have provided objects for most of what is on the boat, but it's not very interesting to look at. We might want to give some more description to these things.

```
"Disenchantment Bay"
```

The Cabin is a room. "The front of the small cabin is entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions, which can be lifted to reveal the storage space underneath. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, and openable. The case is scenery.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin.

The description of the instruments is "Knowing what they do is the Captain's job."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now."

The description of the radar is "Apparently necessary to avoid the larger icebergs."

The description of the radios is "With any luck you will not need to radio for help, but it is reassuring that these things are here."

The order in which we define these things is fairly open. We could also define an object so:

A sign is scenery in the Cabin. The description is "You can get half-day and full-day sight-seeing tours, and half-day and full-day fishing trips."

Where "the description" is assumed to refer to the thing most recently defined, if no object is specified.

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island."

Test me with "examine sign / examine glacier / examine instruments / examine windows / examine radar / examine radios / take the cushions / take the glacier".

These last two commands show how scenery and backdrops are automatically impossible for the player to take.

We mentioned that there is a door out to the deck in our example. The following two sentences will create both the door and the other room:

The cabin door is south of the Cabin and north of the Deck. It is a door and scenery.

Now Inform has constructed a generic room called "Deck" to the south. It has neither a description nor any contents yet, but we could fix that in time. It does have a view of the glacier, though, since we defined the glacier view to be everywhere.

---

23 **Example Disenchantment Bay 6** *WI*

Disenchantment Bay: locking up the charter boat's fishing rods.

It stands to reason that the captain wouldn't let just anyone meddle with his fishing equipment; maybe he keeps that case locked. We could replace the case description with this one, instead:

The Cabin contains a glass case. In the glass case is a collection of fishing rods. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

Now there's a silver key that will unlock it -- though since we haven't said where the key is, the player will never be able to find it in the game. (If we look at the World index, we find "small silver key" right at the bottom, not inside any of the rooms. That is as good as not existing at all -- though we usually use the term "out of play" -- but as we will later see, it is possible to have things initially out of play but brought into existence later on.)

---

25 **Example Disenchantment Bay 7** *WI*

Disenchantment Bay: making the radar and instruments switch on and off.

If we would like the player to be able to turn instrumentation on and off, we could add a line to this effect:

The radar, the instruments, and the radios are devices.

And since the captain is probably not navigating blind, we might also want to say

```
The radar and the instruments are switched on.
```

---

30 **ExampleDisenchantment Bay 9** *WI*
Disenchantment Bay: enter the charter boat's Captain.

Now finally we can put a Captain in the boat:

```
The Captain is a man in the Cabin. "The captain sits at
the wheel, steering the boat and occasionally checking
the radar readout."
```

---

33 **ExampleDisenchantment Bay 10** *WI*
Disenchantment Bay: things for the player and the characters to wear and carry.

At this point we can dress both the Captain and the player with some appropriate props:

```
The captain wears a baseball cap. The description of
the cap is "It says, THE WORST DAY FISHING IS
BETTER THAN THE BEST DAY WORKING."
```

```
The player is carrying a backpack and a bottle of water.
The player is wearing a pair of sunglasses. The
description of the sunglasses is "The light off the water
and the ice does get pretty bright sometimes."
```

(At present the backpack can't be worn, but see the next version.)

If we wanted, we could make the player's backpack infinitely capacious, so:

The backpack is a player's holdall.

...And now whenever the player character is unable to hold everything, he will automatically stow some of his possessions therein.

This is only useful if the player doesn't have infinite carrying capacity himself, so perhaps we also need

The carrying capacity of the player is 3.

Perhaps mercifully, items which are worn are not counted against the player's carrying capacity. We might want to let him take advantage of that, too:

The backpack is wearable.

This capacity system makes a compromise between the realistic and the absurd: on the one hand, it acknowledges that people can't carry an infinite number of items in their hands, while at the same time providing a sack that can.

Many games will have no use for object-juggling of this kind at all; others will want to be much more rigorous about questions of capacity and volume. Fortunately, it is easy to leave the whole business out by assigning no carrying capacity to anything.

We probably do not need a vehicle to ride around our boat, but there might be a heavy ice chest that can only be pushed from room to room:

```
The ice chest is a closed openable container in the
Deck. "A very heavy ice chest sits on the ground." It is
fixed in place and pushable between rooms. A quantity
of ice is in the chest. The description is "Ready and
waiting just in case there's any fish needing to be kept
cool."
```

This anticipates a later chapter, but it would probably be a good idea to hint to the player, if he tries to take the ice chest, that there is another way to move it:

```
Instead of taking the chest: say "It's too heavy to lift, but
you might be able to push it, and just inch it over the
frame of the door."
```

Otherwise, attempts to pick it up will just reply with "That's fixed in place."

Example **Disenchantment Bay 12**

A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

```
"Disenchantment Bay"

Include Locksmith by Emily Short.

Use scoring.

The Cabin is a room. "The front of the small cabin is
```

entirely occupied with navigational instruments, a radar display, and radios for calling back to shore. Along each side runs a bench with faded blue vinyl cushions[if the compartment is closed], which can be lifted to reveal the storage space underneath[otherwise], one of which is currently lifted to allow access to the storage compartment within[end if]. A glass case against the wall contains several fishing rods.

Scratched windows offer a view of the surrounding bay, and there is a door south to the deck. A sign taped to one wall announces the menu of tours offered by the Yakutat Charter Boat Company."

The Cabin contains a glass case. In the glass case is a collection of fishing rods. Understand "rod" as the collection. The case is closed, transparent, openable, lockable, and locked. The case is scenery. The small silver key unlocks the case.

The bench is in the cabin. On the bench are some blue vinyl cushions. The bench is enterable and scenery. The cushions are scenery.

A storage compartment is an openable closed container. It is part of the bench. Instead of opening the bench, try opening the storage compartment. Instead of closing the bench, try closing the storage compartment. Instead of pushing or pulling or turning the cushions, try opening the storage compartment. Understand "space" as the storage compartment.

Some nets and a Coke are in the compartment. Understand "net" as the nets. The description of the nets is "They must have something to do with fish as well. Really, you're just here for the sights." The nets are a container.

Some navigational instruments, some scratched windows, a radar display, and some radios are scenery in the cabin. The radar, the instruments, and the radios are devices. The radar and the instruments are switched on.

A screen is part of the radar. The description of the screen is "[if the radar is switched on]Phantom lights move across the screen.[otherwise]The screen is dark.[end if]". Instead of doing something other than examining to the screen, say "It's not good for much but looking at."

The Captain is a man in the Cabin. "The captain sits at the wheel, steering the boat and occasionally checking the radar readout." The captain wears a baseball cap. The description of the cap is "It says, THE WORST DAY FISHING IS BETTER THAN THE BEST DAY WORKING." The captain carries the silver key. The description of the captain is "[The captain] is wearing [a list of things worn by the captain][if the captain carries something] and carrying [a list of things carried by the captain][end if]."

The description of the instruments is "Knowing what they do is the Captain's job." Instead of doing something other than examining to the instruments in the presence of the Captain: say "The Captain glares at you. Clearly you are not welcome to do that."

The description of the windows is "They're a bit the worse for wear, but you can still get an impressive view of the glacier through them. There were whales earlier, but they're gone now." Understand "window" as the windows.

The description of the radar is "Apparently necessary to avoid the larger icebergs."

The description of the radios is "With any luck you will not need to radio for help, but it is reassuring that these things are here."

A sign is scenery in the Cabin. The description is "You can get half-day and full-day sight-seeing tours, and half-day and full-day fishing trips."

The view of the Malaspina glacier is a backdrop. It is everywhere. The description is "The Malaspina glacier covers much of the nearby slope, and -- beyond it -- an area as large as Rhode Island." Understand "view of the surrounding bay" or "surrounding bay" as the view.

The cabin door is south of the Cabin and north of the Deck. It is a door and scenery. The description of the Deck is "The whole back half of the boat is open, allowing you to view the surroundings without intervening windows -- if you can stand the cold."

The ice chest is a closed openable container in the Deck. "A very heavy ice chest sits on the ground." It is fixed in place and pushable between rooms.

A quantity of ice is in the Deck. "All around the boat bob chunks of glacier ice." Understand "glacier ice" as the quantity. The description is "Curiously cooled into funny-shaped chunks." The printed name of the quantity is "glacier ice".

Instead of taking the quantity of ice when the player is not carrying the nets:
    if the quantity of ice is handled, continue the action;
    say "You are having a hard time fishing out the ice

with your bare hands."

Instead of taking the quantity of ice when the player is carrying the nets:
    if the quantity of ice is handled or the quantity of ice is in the nets, continue the action;
    now the quantity of ice is in the nets;
    say "You scoop up the ice with the net."

Instead of taking the chest: say "It's too heavy to lift, but you might be able to push it, and just inch it over the frame of the door."

The player is carrying a backpack. The player is wearing a pair of sunglasses. The description of the sunglasses is "The light off the water and the ice does get pretty bright sometimes."

The backpack is a player's holdall. The carrying capacity of the player is 3. The backpack is wearable.

Instead of asking the Captain for the key:
    say "'Sure, you can -- well, get me a drink first, would you?'"

Instead of asking the Captain for the key when the Captain is carrying a cold Coke and the Captain is carrying the key:
    move the key to the player;
    say "'Here, knock yourself out.'"

Instead of asking the Captain for the key when the Captain is not carrying the key: say "'I already gave it to you. You didn't lose it, did you?'"

Heat is a kind of value. The heats are cold, cool, room temperature, and warm.

A beverage is a kind of thing. A beverage can be open or closed. A beverage can be openable or unopenable. A beverage is always edible and openable. A beverage has a heat. A beverage is usually warm. The Coke is a beverage. The beer is a beverage. The beer is in the backpack.

Instead of giving or showing a beer to the Captain:
    say "'I don't drink on the job, thanks,' he says. 'You can help yourself if you want it, though.'"

Instead of giving or showing a Coke to the Captain:
    say "'It needs chilling,' the Captain remarks, disgruntled."

Instead of giving or showing a cold Coke to the Captain:
    move the Coke to the Captain;
    increase the score by 2;
    say "'Ah, thank you,' he says. How he can drink an iced soda on a day like today is an open question, but Alaskans are special."

Every turn when the quantity of ice is in the ice chest:
    repeat with item running through beverages in the ice chest:
        let the current heat be the heat of the item;
        if the current heat is not cold, now the heat of the item is the heat before the current heat.

Before printing the name of a beverage (called the drink):

```
    say "[heat of the drink] ".

Understand the heat property as describing a beverage.

The maximum score is 5.

After taking the fishing rods:
    end the story finally;
    increase the score by 3;
    say "Success is yours! (Now if only you knew
anything about fishing.)"

Test me with "test first / test second / test third".

Test first with "x captain / open case / i / ask captain for
the key / give beer to captain / open bench / x nets / get
nets / get coke / give coke to captain".

Test second with "s / open chest / drop nets / get glacier
ice / get nets / get glacier ice / g / put glacier ice in chest
/ get coke / put coke in chest".

Test third with "x coke / g / g / g / get coke / n / give
coke to captain / ask captain for key / open case / get
rods".
```

Example **About Inform's regular expression support**

Some footnotes on Inform's regular expressions, and how they compare to those of other programming languages.

There is not really any unanimity about what regular expression language is. The unix tools sed and grep extend on Kleene's original grammar. Henry Spencer's regex library extended on this again, and was a foundation for Perl, but Perl once again went further. Philip Hazel's PCRE, despite the name Perl Compatible Regular Expressions, makes further extensions still, and so on.

Inform's regular expressions are modelled on those of Perl, as the best de facto standard we have, but a few omissions have been inevitable. Inform's regex matcher must occupy source code less than one hundredth the size of PCRE, and it has very little memory. Inform aims to behave exactly like Perl except as follows:

(i) Inform allows angle brackets as synonymous with square brackets, for reasons explained above. This means literal angle brackets have to be escaped as "\<" and "\>" in Inform regular expressions, which is unnecessary in Perl.

(ii) Inform only has single-line mode, not multiline mode: this removes need for the mode-switches "(?m)" and "(?s)" and the positional markers "\A" and "\Z". Multiline mode is idiosyncratic to Perl and is a messy compromise to do with holding long files of text as single strings, yet treating them as lists of lines at the same time: this would not be sensible for Inform. Similarly, because there is no ambiguity about how line breaks are represented in Inform strings (by a single "\n"), initial newline convention markers such as "(*ANYCRLF)" are unsupported.

(iii) The codes "\a", "\r", "\f", "\e", "\0" for alarm, carriage return, form feed, escape and the zero character are unsupported: none of these can occur in an Inform string.

(iv) Inform does not allow characters to be referred to by character code (whereas Perl allows "\036" for an octal character code, "\x7e" for a hexadecimal one, "\cD" for a control character). This is because we do not want the user to know whether text is internally stored as ZSCII

or Unicode.

(v) Inform's character class "\p" (and its negation "\P") have no equivalent in Perl, and Inform's understanding of "\w" is different. Perl defines this as an upper or lower case English letter, underscore or digit, which is good for programming-language identifiers, but bad for natural language - for instance, "é" is not matched by "\w" in Perl, but unquestionably it appears in words. Inform therefore defines "\w" as the negation of "\s" union "\p".

(vi) Inform supports only single-digit grouping numbers "\1" to "\9", whereas Perl allows "\10", "\11", ...

(vii) POSIX named character ranges are not supported. These are only abbreviations in any case, and are not very useful. (Note that the POSIX range "[:punct:]", which is supposedly for punctuation, includes many things we do not want to think of that way - percentage signs, for instance - and so "\p" has a more natural-language-based definition.)

(viii) Character classes can be used inside ranges, so that "<\da-f>" is legal, but not as ends of contiguous runs, so that "<\d-X>" is not legal. (As reckless as this is, it is legal in Perl.)

(ix) For obvious reasons, escapes to Perl code using the notation "(?{...})" are unsupported, and so is the Perl iteration operator "\G".

(x) Perl's extended mode "(?x)", a lexical arrangement which allows expressions to be expanded out as little computer programs with comments, is unsupported. It would look awful syntax-coloured in the Inform interface

and is not a style of coding to be encouraged.

Inform further does not support the Python extension of named subexpression groups, nor the Java extension of the possessive quantifier "++". There was only so much functionality we could squeeze in.

As verification of Inform's matching algorithm, we took the Perl 5 source code's notorious "re-test.txt" set of 961 test cases, removed the 316 using features unsupported by Inform (220 tested multiline mode, for instance), and ran the remaining 645 cases through Inform. It agrees with Perl on 643 of these: the two outstanding are -

(i) Perl is able to match "^(a\1?){4}$" against "aaaaaa" but Inform is not - Inform's backtracking is not as good when it comes to repetitions of groupings which are recursively defined. (Note that the optional "\1" match refers to the value of the bracketed expression which contains it, so that the interpretation is different on each repetition. Here to match we have to interpret "?" as 0, 0, 1, 0 repeats respectively as we work through the "{4}".)

(ii) Perl matches "((<a-c>)b*?\2)*" against "ababbbcbc" finding the match "ababb", whereas Inform finds the match "ababbbcbc". This is really a difference of opinion about whether the outer asterisk, which is greedy, should be allowed three matches rather than two if to do so requires the inner asterisk, which is not greedy, to eat more than it needs on one of those three matches.

Case (i) is a sacrifice to enable Inform's back-tracking to use less memory. Case (ii) simply seems unimportant.

Example **Formal syntax of sentences**

A more formal description of the sentence grammar used by Inform for both assertions and conditions.

An entire grammar for the whole mass of Inform would not be linguistically interesting: it contains many convenient wordings which are not really part of a grand pattern. Inform does, however, have a formal

notion of a Sentence, a grammatical structure which we shall call S. It is almost true that conditions ("if the flowerpot is on the wall") have the same grammar as assertions ("The flowerpot is on the wall") and "now" phrases ("now the flowerpot is on the wall"). All three use the S grammar, so we could define an assertion as "S.", say that "if S", "while S", "when S" and so on are conditions, and say that "now S" defines the "now" declaration.

Grammatical sentences do not necessarily make sense, of course. Many perfectly grammatical assertions in fact give rise to problem messages:

> The wicker basket is not in the kitchen. (*Unhelpfully negative.*)
> The wicker basket has been in the kitchen. (*Talks about a time which never existed.*)
> The wicker basket is full. (*Full of what? Too vague.*)
> The wicker basket is the ginger cat. (*Demonstrably false.*)

Whereas the first three, at least, would be sensible as conditions. So saying that assertions are "just like" conditions is a little misleading: what they have in common is S, the underlying grammar they each use as a starting-point.

To define S, we break it up into subsidiary structures. The most important is the Description Phrase (DP), examples of which include "the red basket", "somewhere lighted" and "an empty open container". Clearly sentences include DPs, but they also include other ingredients. The general pattern used in Inform is very simple:

> 1. S = DP + VP
> 2. VP = Verb + DP

where VP is another structure, the Verb Phrase. For instance:

> S (The horseman wears a riding helmet)
> = DP (The horseman) + VP (wears a riding helmet)
> VP (wears a riding helmet)
> = Verb (wears) + DP (a riding helmet)

In that example, the Verb was the single word "wears". More generally, Inform allows a Verb to include adverbs and prepositions, to be negated, and to come in any of four tenses, so the following are all valid examples

of Verb in our grammar:

wore
carries
is carried by
had not been inside

Although we are not going through the definition of Description Phrases in detail, it is worth noticing how "which" and "who" behave:

3a. DP = DP + which + VP
3b. DP = DP + who + VP

Thus "an open container which is in the Ballroom" can be broken down as:

DP (an open container) + which + VP (is in the Ballroom)

To understand compounds like "something in a container", we have to invent a new grammatical structure for "in a container" and similar: let's call this a Relative Phrase (RP).

4. DP = DP + RP

Thus "an open container in the Ballroom" is DP (an open container) + RP (in the Ballroom). Relative Phrases have two different forms:

5a. RP = Preposition + DP
5b. RP = Participle + DP

so that "in a container" is an example of 5a. An example of 5b would be

RP (worn by Mr Darcy) = Participle (worn by) + DP (Mr Darcy)

That is nearly it, but not quite: we must go back to the "almost" in the

statement above that assertions and conditions "almost" have the same grammar S. The difference arises from a curious irregularity in English called subject-verb inversion (see the *Oxford English Grammar* at 3.22F), whereby assertions can be reversed but not conditions. For instance,

In the Garden is a sunflower.

This does not follow the pattern S = DP + VP, because "in the garden" is not a DP: indeed, it is not a noun at all. To make sense of this sentence, Inform reverses it to "A sunflower is in the Garden", which does indeed follow DP + VP. Hence the final rule:

6 (assertions only). S = RP + Verb + DP

So the condition "if in the garden is a sunflower..." fails because rule 6 does not apply to the grammar for conditions: while occasional poetic uses of subject-verb inversion do turn up in conditions ("If On A Winter's Night A Traveller", say), they are rare in ordinary English usage, and illegal in Inform. That completes the S grammar, so to recap:

1. S = DP + VP
2. VP = Verb + DP
3a. DP = DP + which + VP
3b. DP = DP + who + VP
4. DP = DP + RP
5a. RP = Preposition + DP
5b. RP = Participle + DP
6 (assertions only). S = RP + Verb + DP

Example **Mathematical view of relations** *WI*

Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

Inform uses the term "relation" in a broader sense than mathematics. Properly speaking, the term "relation" in its mathematical sense only applies to the case where the domain for the left and right objects are the same: for simplicity's sake, let us talk only about the case where they are.

In mathematics, the properties most often looked for in a relation are that it should be:

(a) Reflexive: A <=> A for every A. This is not especially useful for Inform, and seldom appears in practical examples.

(b) Symmetric: A <=> B if and only if B <=> A. Generally, Inform relations are not symmetric, but there are two important cases which are:

```
Meeting relates people to each other.
Marriage relates one person to another.
```

These are automatically symmetric, so that to assert one way round is to assert the other as well.

(c) Transitive: A <=> B and B <=> C means that A <=> C as well. Again, Inform relations are not generally transitive. In many relations, there can be long chains of things, each perhaps related to the one in front and the one behind, so that there is some indirect sense in which the two ends of the chain are connected to each other: but they are not related as such. For instance, a journey across the map might pass through ten rooms, each adjacent to the last and next, but the two ends would not themselves be adjacent. The concept we need is the "transitive closure" of the original relation, defined as the smallest transitive relation including the original. If R is a relation between "things", then the following:

```
TC relates a thing (called A) to a thing (called B) when
the number of steps via R from A to B is greater than 0.
```

is the transitive closure of R. In particular,

```
Accessibility relates a room (called A) to a room (called
B) when the number of moves from B to A is greater
than 0. The verb to be accessible from means the
accessibility relation.
```

calculates the transitive closure of adjacency. Here, though, the way we normally understand "accessible from" suggests that it would be better to write:

```
Accessibility relates a room (called A) to a room (called
```

B) when the number of moves from B to A is at least 0.

which is reflexive as well as transitive. The usefulness of Inform's "next step via R from A to B" construction, in a wide variety of settings, reflects the importance of transitivity as an idea.

A relation which has all three properties of being reflexive, symmetric and transitive is called an "equivalence relation". (If all the map connections are two-way, then the accessibility relation above is symmetric and therefore a full equivalence relation: but if not, it may not be.) Inform has a special construction for making equivalence relations:

Nationality relates people to each other in groups.

This language - "in groups" - relies on the standard theorem that every equivalence relation on a set naturally defines a partition of that set, and vice versa. The "groups" referred to are what are normally called "equivalence classes". (Inform does little with these equivalence classes: it might be interesting to do so, in effect forming quotient kinds.)

---

244 **Example** **Graph-theory view of relations** *WI*

Some notes on relations from the point of view of graph theory.

One way to look at a relation is to regard it as a directed graph: that is, a collection of things ("vertices") with arrows drawn between them ("edges"). We write our items A, B, C, ... on a piece of paper: then, if A relates to B, we draw an arrow pointing from A to B, and so on. If we made this drawing for the adjacency relation, we would more or less have reconstructed the map (or at least a simplified one which does not care about precise directions, like the famous diagram of the London Underground). But the drawing can be made for any relation. If we define:

Suspecting relates various people to one person.

then, in the corresponding graph, each "vertex" will have at most one arrow leading away from it - though there could be many (or none) leading towards. Conversely, a one-to-various relation produces a graph where each vertex has at most one arrow coming in. A one-to-one relation means that the picture consists of some vertices on their own,

with no arrows, a few perhaps with looped arrows leading from and to themselves, and then a collection of pairs joined by arrows. On the other hand, a various-to-various relation is just a free-for-all, with no restrictions on the arrows. The relations:

```
Meeting relates people to each other.
Marriage relates one person to another.
```

always have the property of working both ways round, and these are easiest to visualise by forgetting the direction of the arrows, so that they just become lines joining the vertices.

Inform uses a different algorithm for finding routes ("the next step via R from A to B") in each of these cases, and internally it stores relations in different formats in the different cases, because it makes a big difference to the efficiency of Inform to minimise the storage required for a relation and the time taken to explore it.

All the cases are benign except for "various to various" - the most useful - and for its closely related symmetrical version, "relates... to each other". Inform cannot afford to assume that the relation will be "sparse" (that is: that no vertex will have more than a certain number of arrows, or that the total number of arrows will be small), because it has no idea how the arrows will come and go during play. It therefore uses 1 bit of storage for each pair of objects. This sounds harmless, but if there are 200 rooms, there are 40,000 pairs of rooms, which means a 5000-byte allocation of storage (plus a handful of bytes as overhead). Gratuitous various-to-various relations are therefore not a good idea.

There is a standard algorithm for calculating shortest paths through a directed graph, but Inform does not always use it, because there is not always memory to store the required matrix of partial results. Inform's slow method, likely to be used on the Z-machine, requires a storage overhead which is equal to the number of vertices, not the square of that number, but the worst-case running time can be bad: if there are N vertices, and the diameter of graph (the longest distance between vertices) is D, then the running time is proportional to D times N. The worst case in finding routes from A to B is when almost every vertex can reach B, some across long trails, but A cannot. In the case of finding routes across the game's map, this must be multiplied further by the number of possible directions - usually 16.

This does not sound too awful, but if one is trying to find (say) "the most distant room from A", that means a further loop and now the running time will be D times N squared. Extension writers will need to be careful of this kind of thing: it is easy to write highly cool prototypes which work terribly slowly on larger, more realistic maps.

Example **Backus-Naur form for rules** *WI*

The full grammar Inform uses to parse rule definitions, in a standard computer-science notation.

Backus-Naur form, or BNF, is a standard notation used by computer scientists to specify more or less precisely what the valid programs are for a given programming language. It tends to provide a good description for a language such as C or Pascal, where contextual rules are limited, but the authors of Inform are doubtful that it is such a good tool for a natural-language system. For those who are interested, though, the following gives a formal specification for Inform's rules.

```
<rule> ::=
    Definition : A/an <kind> is <new adjectival name> if/
unless <definition>
    | <preamble> : <phrases>
    | <preamble> , <phrase> (* only allowed for a few
cases: see below)



<definition> ::=
    <condition>
    | its/his/her/their <value property name> is/are
<value> or less/more
    | : <phrases>



<preamble> ::=
    To <phrase template>
    | To decide if/whether <phrase template>
    | To decide which/what <kind of value> is <phrase
template>
    | This is the <rule name>
    | [[A] Rule for] <circumstances> [(this is the <rule
name>)]



<circumstances> ::=
    At <time>
    | When <event name>
```

```
    | [<placement>] <rulebook reference> [while/when
<condition>] [during <scene name>]



<rulebook reference> ::=
    <rulebook name> [about/for/of/on/rule] [<action
pattern>]
    | <object-based-rulebook name> [about/for/of/on/rule]
[<description>]



<placement> ::=
    a/an
    | [the] first
    | [the] last



<phrases> ::=
    <phrase>
    | <phrases> ; <phrase>
```

The following examples show how Inform breaks down some typical rules using the system above:

```
<rule> = At 2:09 PM: increase the score by 2; say
"Progress!"
    <preamble> = At 2:09 PM
        <circumstances> = At 2:09 PM
            At
            <time> = 2:09 PM
    :
    <phrases> = increase the score by 2; say "Progress!"
        <phrase> = increase the score by 2
        ;
        <phrase> = say "Progress"



<rule> = Instead of eating the ostrich during Formal
Dinner (this is the cuisine rule), say "It's greasy!"
    <preamble> = Instead of eating the ostrich during
Formal Dinner (this is the cuisine rule)
        <circumstances> = Instead of eating the ostrich
during Formal Dinner
```

```
            <rulebook reference> = Instead of eating the
ostrich
               <rulebook name> = Instead
               of
               <action pattern> = eating the ostrich
            during
            <scene name> = Formal Dinner
         (
         this
         is
         the
         <rule name> = cuisine rule
         )
      ,
      <phrases> = say "It's greasy!"
         <phrase> = say "It's greasy!"



<rule> = After printing the name of a container: say "!"
   <preamble> = After printing the name of a container
      <circumstances> = After printing the name of a
container
            <rulebook reference> = After printing the name
of a container
               <object-based-rulebook name> = After
printing the name
               of
               <description> = a container
   :
   <phrases> = say "!"
      <phrase> = say "!"
```

(*) The colon dividing a rule preamble from its definition can be replaced by a comma only if the preamble begins with the words "Instead of", "Before", "After", "Every turn" or "When", and if the definition consists only of a single phrase.

Chapter 2: Adaptive Prose

§2.1. Varying What Is Written; *§2.2. Varying What Is Read*; *§2.3. Using the Player's Input*

Contents of *The Inform Recipe Book*
Chapter 1: How to Use The Recipe Book
Chapter 3: Place
Indexes of the examples

§2.1. Varying What Is Written

Before getting to actual recipes, many recipe books begin with intimidating lists of high-end kitchen equipment (carbon-steel pans, a high-temperature range, a Provencal shallot-grater, a set of six pomegranate juicers): fortunately, readers who have downloaded Inform already have the complete kitchen used by the authors. But the other traditional preliminaries, about universal skills such as chopping vegetables, boiling water and measuring quantities, do have an equivalent.

For us, the most basic technique of IF is to craft the text so that it smoothly and elegantly adapts to describe the situation, disguising the machine which is never far beneath the surface. This means using text substitutions so that any response likely to be seen more than once or twice will vary.

M. Melmoth's Duel demonstrates three basic techniques: an ever-changing random variation, a random variation changing only after the player has been absent for a while, and a message tweaked to add an extra comment in one special case. (Random choices can be quite specifically constrained, as **Ahem** shows in passing.) **Fifty Ways to Leave Your Larva** and **Fifty Times Fifty Ways** show how a generic message can be given a tweak to make it a better fit for the person it currently talks about. **Curare** picks out an item carried by the player to work into a message, trying to make an apt rather than random choice. **Straw Into Gold** demonstrates how to have Inform parrot back the player's choice of name for an object.

Another reason to vary messages is to avoid unnatural phrasing. **Ballpark** turns needlessly precise numbers - another computerish trait - into more idiomatic English. (Likewise **Numberless**, though it is really an example demonstrating how to split behaviour into many cases.) **Prolegomena** shows how to use these vaguer quantifiers any time Inform describes a group of objects (as in "You can see 27 paper clips here.").

Blink, a short but demanding example from the extreme end of *Writing with Inform*,

shows how the basic text variation mechanisms of Inform can themselves be extended. **Blackout** demonstrates text manipulation at a lower level, replacing every letter of a room name with "*" when the player is in darkness.

Inform's included extension Complex Listing allows us more control over the order and presentation of lists of items.

For how to change printed text to upper, lower, sentence, or title casing, see **Rocket Man**.

Start of Chapter 2: Adaptive Prose

Back to Chapter 1: How to Use The Recipe Book: §1.4. Information Only

Onward to §2.2. Varying What Is Read

Example 71: ★ **Fifty Ways to Leave Your Larva** Using text substitution to make characters reply differently under the same circumstances.

Example 169: ★ **Ahem** Writing a phrase, with several variant forms, whose function is to follow a rule several times.

Example 174: ★ **Numberless** A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

Example 178: ★ **M. Melmoth's Duel** Three basic ways to inject random or not-so-random variations into text.

Example 249: ★ **Olfactory Settings** Some adaptive text for smelling the flowers, or indeed, anything else.

Example 252: ★ **Responsive** Altering the standard inventory text for when the player is carrying nothing.

Example 343: ★ **Prolegomena** Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

Example 345: ★ **Wesponses** Parser messages that are delivered with a speech impediment.

Example 413: ★ **Rocket Man** Using case changes on any text produced by a "to say..." phrase.

Example 419: ★ **Blackout** Filtering the names of rooms printed while in darkness.

Example 439: ★ **Curare** A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

Example 454: ★ **Blink** Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

Example 245: ★★ **Fun with Participles** Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on

Clark's idle activity.

Example 246: ★★ **Variety** Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

Example 247: ★★ **Variety 2** This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

Example 250: ★★ **History Lab** We create phrases such as "the box we took" and "the newspaper Clark looked at" based on what has already happened in the story.

Example 251: ★★ **Relevant Relations** An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

Example 60: ★★★ **Ballpark** A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

Example 72: ★★★ **Fifty Times Fifty Ways** Writing your own rules for how to carry out

substitutions.

Example 248: ★★★ **Narrative Register** Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

Example 305: ★★★ **Straw Into Gold** Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or "man" -- depending on which the player last used -- until the first time the player refers to him as "Rumpelstiltskin".

---

§2.2. Varying What Is Read

Making the printed text adapt to circumstances only makes half of the conversation graceful: the other half is to allow the player's commands to have a similar freedom. The things the player can refer to should always respond to the names which would seem natural to the player. Inform provides a variety of techniques for understanding words always, or only under certain conditions; and, if need be, we can also get direct access to what the player has typed in order to examine it with regular expressions. (This last resort is rarely necessary.)

First Name Basis shows how to assign names to things or to kinds of thing - if, for instance, we want the player to be able to refer to any man as "man" or "gentleman":

```
Understand "man" or "gentleman" as a man.
```

We may also sometimes want to give names that are specifically plural, as in

```
A duck is a kind of animal. Understand "birds" as the plural of duck.
```

or

```
Understand "birds" as the plural of the magpie.
```

Vouvray demonstrates.

A common challenge arises when two objects have names that overlap or are related, and we wish Inform to choose sensibly between them: for instance, a cigarette vs. a cigarette case. If a word should apply to something only as part of a phrase (e.g., "cigarette" alone should never refer to the cigarette case) we can manage the situation as follows:

```
The case is a closed openable container. The printed name is
"cigarette case". Understand "cigarette case" as the case.
```

Because "cigarette" here appears only as part of the phrase "cigarette case", it will be understood only in that context; the conflict with the bare cigarette will not arise.

As a variant, we may want one object only to take precedence over another in naming. If we wanted the player to be allowed to refer casually to the cigarette case as "cigarette" when (and only when) the cigarette itself is not in view, we could add

```
Understand "cigarette" as the case when the cigarette is not visible.
```

Tricks which consider the visibility of other objects can be bad for performance if used widely; but for adding finesse to the treatment of a few items, they work very well.

(There may still arise cases where the player uses a name which can legitimately refer to two different things in view. To deal with this situation, we may want the Does the player mean... rules, explained in the chapter on Understanding; and to change the way the story asks for clarification, see the two activities Asking which do you mean and Clarifying the parser's choice of something.)

Names of things which contain prepositions can also be tricky because Inform misreads the sentences creating them: **Laura** shows how some awkward cases can be safely overcome.

A more difficult case is to ensure that if we change the description or nature of something in play, then the names we understand for it adapt, too. "Understand... when..." can be all that's needed:

```
Understand "king" as Aragorn when we have crowned Aragorn.
```

Or, similarly, if we want some combination of categories and characteristics to be recognized:

```
Understand "giant" as a man when the item described is tall.
```

"The item described" here refers to the thing being named. "...when" can even be useful in defining new commands, and **Quiz Show** demonstrates how to ask open-ended questions that the player can answer only on the subsequent turn.

Properties can also be matched without fuss:

```
Tint is a kind of value. The tints are green, aquamarine and darkish
purple. The wallpaper is fixed in place in the Hotel. The wallpaper
has a tint. Understand the tint property as describing the wallpaper.
```

This allows EXAMINE AQUAMARINE WALLPAPER if, but only if, it happens to be aquamarine at the moment. Relationships can also be matched automatically:

```
A box is a kind of container. The red box is a box in the Toyshop.
Some crayons are in the red box. Understand "box of [something
related by containment]" as a box.
```

which recognises BOX OF CRAYONS until they are removed, when it reverts to plain BOX only.

Greater difficulty arises if, using some variable or property or table to mark that a bottle contains wine, we print messages calling it "bottle of wine". We are then honour-bound to understand commands like TAKE BOTTLE OF WINE in return, not to insist on TAKE BOTTLE. Almost all "simulation" IF runs in to issues like this, and there is no general solution because simulations are so varied.

A converse challenge arises when we want to *avoid* understanding the player's references to an object under some or all circumstances. This is relatively uncommon, but does sometimes occur. For this situation, Inform provides the "privately-named" property, as in

```
The unrecognizable object is a privately-named thing in the Kitchen.
```

Here "privately-named" tells Inform not to understand the object's source name automatically. It is then up to us to create any understand lines we want to refer to the object, as in

```
Understand "oyster fork" as the unrecognizable object when the
etiquette book is read.
```

Of course, if we need an object that the player is never allowed to refer to at all, we can just make this privately-named and then not provide any understand lines at all.

A final source of difficulty is that by default Inform truncates words to nine letters before attempting to identify them. This is no problem in most circumstances and is likely to go unnoticed -- until we have two very long words whose names are nearly identical, such as "north-northwest exit" and "north-northeast exit". (To make matters worse, a punctuation mark such as a hyphen counts as two letters on its own.)

When we are compiling for Glulx, the limit is easily changed with a single line, setting the constant called DICT_WORD_SIZE. For instance, if we wanted to raise the limit to 15, we would write

```
Use DICT_WORD_SIZE of 15.
```

When compiling for the Z-machine, the solution is harder. **North by Northwest** shows how to use the reading a command activity to pre-process very long names, rendering them accessible to the parser again.

Inform also allows the player to refer to the most recently seen objects and people as IT, HIM, HER, and so on. It sets these pronouns by default, but there are times when we wish to override the way it does that. **Pot of Petunias** shows off a way to make Inform recognize an object as IT when it would not otherwise have done so.

See **Liquids** *for a resolution of this bottle-of-wine issue*

See **Using the Player's Input** *for an example (Mr. Burns' Repast) in which a fish can be called by any arbitrary word as long as it ends in the letters -fish*

See **Memory and Knowledge** *for a way to refer to characters whom the player knows about but who aren't currently in the room*

See **Clarification and Correction** *for ways to improve guesses about what the player means*

See **Alternatives To Standard Parsing** *for several esoteric variations on the default behavior, such as accepting adverbs anywhere in the command, and scanning the player's input for keywords*

Back to §2.1. Varying What Is Written

Onward to §2.3. Using the Player's Input

Example 11: ★ **First Name Basis** Allowing the player to use different synonyms to refer to something.

Example 323: ★ **Quiz Show** In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

Example 325: ★ **Pot of Petunias** Responding sensibly to a pot of petunias falling from the sky.

Example 19: ★★ **Laura** Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

Example 43: ★★ **Vouvray** Adding synonyms to an entire kind of thing.

Example 375: ★★ **North by Northwest** Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed

words.

§2.3. Using the Player's Input

We may sometimes want to capture specific words the player has used and then feature that text elsewhere in the story.

Terracottissima Maxima demonstrates using text to describe objects; **Mr. Burns' Repast** lets the player refer to a fish by any of a number of names, and changes the way the fish is described as a result.

More specialized effects are also possible: **Xot** shows how to collect the player's erroneous input and store the command line to be printed back later. **Igpay Atinlay** shows how to parrot the player's command back in pig Latin form.

See **Animals** *for a dog which the player can re-name*

See **Traits Determined By the Player** *for a way to let the player name the player character*

Start of Chapter 2: Adaptive Prose

Back to §2.2. Varying What Is Read

Onward to Chapter 3: Place: §3.1. Room Descriptions

Example 421: **Igpay Atinlay** A pig Latin filter for the player's commands.

Example 422: **Mr. Burns' Repast** Letting the player guess types for an unidentifiable fish.

Example 317: **Terracottissima Maxima** Flowerpots with textual names that might change during play.

Example 381: **Xot** Storing an invalid command to be repeated as text later in the game.

Examples from Chapter 2: Adaptive Prose

Start of this chapter
Chapter 3: Place
Indexes of the examples

Example **Fifty Ways to Leave Your Larva** *WI*
Using text substitution to make characters reply differently under the same circumstances.

We can use these substitutions to put together fairly complicated variations within a single piece of text:

```
"Fifty Ways to Leave Your Larva"

The Beekeeper's Palace is a room. Wasp is a woman in
the palace. Drone is a man in the palace.

Instead of kissing someone:
    say "'[denial], [insult]! [boast]!'";
```

In this context, [denial] is understood to refer to the denial property of the noun -- but we could spell it out with "denial of the noun" if we wanted to.

```
A person has some text called denial. The denial of a
person is usually "Stand back". The denial of Drone is
"You forget yourself"

A person has some text called insult. The insult of a
person is usually "Grasshopper". The insult of Wasp is
"Larva".
```

```
A person has some text called boast. The boast of a
person is usually "I am ferocious". The boast of Drone is
"I have ferocious allies".
```

And then it would be trivial to insert further rules using these responses:

```
Instead of attacking someone:
    say "'Get away, [insult]!'"


Test me with "kiss wasp / hit wasp / hit drone / kiss
drone".
```

---

169 ExampleAhem *WI*

Writing a phrase, with several variant forms, whose function is to follow a rule several times.

As we see in the example here, it is possible to use slashed variations in more than one place in a phrase, and to offer a number of separate forms. The main rule of thumb to remember is that value inputs for the phrase should always be separated by some text; so

```
To do/follow (chosen rule - a rule) exactly/precisely/
just/-- (N - a number) time/times:
    ....
```

would cause a problem when we tried to call it with

```
follow the throat-clearing rule 2 times.
```

In general, we probably don't need to make our phrase definitions quite so flexible as this, but it's a good idea to account for "a" vs. "the", and for the possibility of using singular and plural forms, especially when writing extensions or other source to be shared.

```
"Ahem"

To do/follow (chosen rule - a rule) exactly/precisely/just
(N - a number) time/times:
    repeat with index running from 1 to N:
        follow chosen rule.

This is the throat-clearing rule:
    say "'Ahem,' says [a random visible person who is not
the player]."

After waiting:
    do the throat-clearing rule just one time.

Instead of listening:
    follow the throat-clearing rule precisely three times.

Instead of smelling:
    follow the throat-clearing rule exactly 2 times.

Chateau Marmont is a room. Tom, Jack, Zsa-Zsa, and
Wilma-Faye are people in the Chateau. Zsa-Zsa and
Wilma-Faye are women.

Test me with "wait / smell / listen".
```

ExampleNumberless

A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

```
"Numberless"


The Rambling Warren is a room.


When play begins:
    let N be a random number between 1 and 5;
    if N is 1:
        say "N is one.";
    otherwise if N is 2:
        say "N is two.";
    otherwise if N is 3:
        say "N is three.";
    otherwise:
        say "N is more than the number of your toes."
```

The final "otherwise" here will fire only if none of the earlier conditions applies; we could leave it out and print nothing in the case that N is 4 or 5.

The more compact way to do this is to create a list of values that our number could match; in many programming languages this is called a switch statement. For example:

```
When play begins:
    let Y be a random number between 6 and 10;
    if Y is:
        -- 6: say "Six is the magic number!";
        -- 7: say "The number of the day is seven!";
        -- otherwise: say "Today's magic number is boring."
```

As a final option, we can use a construction we've seen only briefly before now: a table. The use of tables will be explained more fully in their own chapter, but here we see in brief that we can assign a number of values to one column of a table and then use that table to look up output:

```
When play begins:
    let X be a random number between 11 and 14;
    if X is a number listed in the Table of Switching, say
"[output entry][paragraph break]";
```

otherwise say "X is greater than the number of your noses!"

Table of Switching

output number
"X is eleven!"
"X is twelve!"
"X is thirteen!"

Test me with "z".

As we shall see, things other than text can be stored in tables, so we could also use a table as a way to look up objects or even rules to carry out.

Example M. Melmoth's Duel

Three basic ways to inject random or not-so-random variations into text.

"M. Melmoth's Duel"

Saint-Germain-des-Prés is a room. "Haunt of artists, of the coffee-drinking sort, and of cafés, of the artist-haunted sort, you once again find yourself outside M. Melmoth's hotel. Today [one of]the recently-fallen rain runs down the gutters of the 6th[or]sunlight glints even off the blackened windows of the Abbey[or]crowds of vulgar children play chase around the lampposts[at random], and you long to be indoors."

The Hôtel d'Alsace is inside from Saint-Germain-des-Prés. "Typical. Oscar writes you a letter announcing his

own imminent demise - 'My wallpaper and I are fighting a duel to the death. One or other of us has got to go.' - and then you get there and he's out, no doubt procuring paint the colour of absinthe, if he isn't procuring the painter."

Tint is a kind of value. The tints are green, aquamarine and darkish purple.

The wallpaper is fixed in place in the Hôtel. The wallpaper has a tint. "In this light, the wallpaper has a distinctly [tint of the wallpaper] wash. [if the tint of the wallpaper is darkish purple]You particularly dislike purple.[end if]"

Before going to the Hôtel: now the wallpaper is a random tint.

After going from the Hôtel, say "You leave, shaking your head. But within twenty-four hours, you are back, as you always knew you would be."

Test me with "in / out / look / in / out / look".

ExampleOlfactory Settings

Some adaptive text for smelling the flowers, or indeed, anything else.

While this isn't very interesting as IF, it runs through most of the adaptive-text tricks.

"Olfactory Settings"

The Doghouse is a room. "Not so much a place as a state of being."

The player carries a ticket to the opera, some papers, and a bouquet of flowers. The bouquet is ambiguously plural.

Instead of eating something inedible, say "[The noun] [don't] seem likely to agree with [us] at all. [We]['d be wiser to leave [regarding the noun][them] alone."

Instead of touching something: say "[regarding the noun][Those] [are] all prickly."

Instead of smelling something: say "[Our] nose [regarding nothing][are] too weak to get much smell from [regarding the noun][those]."

Instead of smelling the bouquet: say "[regarding the noun][They]['re] lovely."

Instead of tasting something:
    say "Whew, [regarding the noun][are] [those] ever nasty!"

Test me with "x ticket / eat it / eat them / touch it / touch them / smell it / smell them / taste it / taste them / x papers / eat it / eat them / touch them / smell them / taste them / x bouquet / eat it / eat them / touch them / smell them / taste them".

252 Example **Responsive**

Altering the standard inventory text for when the player is carrying nothing.

WI

The most straightforward way to alter the response text for something in the standard rules is to select the Index tab, then select Actions, then pick the particular action whose text we'd like to alter. Under action details, there will be icons that look like speech bubbles.

Clicking on the speech bubble will show what the current response text is, and give us an option called "set".

If we click "set", this will automatically paste in the response name that we need to change. We can put this inside a "when play begins" rule in order to make that change take effect from the start of the game, like so:

```
"Responsive"

An Anonymous B613 Cell is a room. "There isn't much
to see in this bare room. What there is, you've already
seen sometime in the last twenty years."

When play begins:
    now print empty inventory rule response (A) is "[We]
[have] absolutely nothing.".

Test me with "i".
```

343 Example **Prolegomena**

Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

WI

Room descriptions often make the player character out to be a bit of a

savant, able to count whole stacks of items at a glance: "You see 27 paper clips here."

We can adjust this behavior to our liking, though, with the printing a number... activity, as follows:

```
"Prolegomena"


The Editor's Office is a room. The desk is a supporter in
the Editor's Office.


A red pencil is a kind of thing. 12 red pencils are on the
desk.


A letter is a kind of thing. 12 letters are on the desk.
Understand "correspondence" as a letter.


Rule for printing the plural name of a letter:
    if the listing group size is greater than 7, say
"correspondence";
    otherwise say "letters".


Rule for printing a number of something (called the
target) when the listing group size is greater than 7:
    say "[one of]some [or]various [or]an assortment of [at
random]";
    carry out the printing the plural name activity with the
target.
```

This general rule can of course be overridden by more specific ones; for instance, if we want to take the opportunity to comment on the viewpoint character's appetite for instruments of correction:

```
Rule for printing a number of red pencils (called the
```

```
target) when the listing group size is greater than 10:
    carry out the printing the plural name activity with the
target;
    say " in nearly-sufficient quantity".



Test me with "get two letters / look / get a pencil / i / get
pencil / g / g / look / i / get all / i".
```

ExampleWesponses

Parser messages that are delivered with a speech impediment.

If we want to change individual responses to player action, then the best thing to do is to use the response facility to modify those selections, as shown in the chapter on Responses.

However, suppose what we want is to give the parser a speech impediment that slightly alters all of the responses it issues. For this purpose, we might need to do a bit of text replacement...

```
"Wesponses"



The Office is a room. Barry Kripke is a man in the
Office.



The response inhibition is initially false.



Rule for issuing the response text of a response (called
R) when response inhibition is false:
    now response inhibition is true;
    let output be "[text of R]";
    now response inhibition is false;
    replace the text "r" in output with "w";
    replace the text "R" in output with "W";
```

```
    say "[output]".

Test me with "i / x barry / listen / waffle / jump"
```

Notice that this doesn't affect the printed names of objects in the room description or other kinds of output text -- only those that are issued by the response mechanism.

It would also not work to try to give our parser a nervous personality by simply adding "Um, " to the beginning of each response, because responses are not guaranteed to be full standalone sentences. For example, we could imagine writing

```
Rule for issuing the response text of a response (called
R) when response inhibition is false:
    now response inhibition is true;
    let output be "Um, [text of R]";
    now response inhibition is false;
    say "[output]".
```

but here is what the room description would say as a result:

```
Office
Um, You Um, can see Barry KripkeUm, here
```

---

413 **Example** **Rocket Man** *WI*

Using case changes on any text produced by a "to say..." phrase.

We can now change the case of any text produced by a "to say..." phrase. This is often useful when we would like to make use of a standard say phrase in some new context. Say, for instance, that we would like to "[is-are the list...]" in a context that needs the first letter to be capitalized.

We could write a new say phrase, such as "to say is-are the list of (N - a description of objects) in sentence capitalization"; but there is an easier way, and that is to set a text variable to the output of the to say phrase, and then print that text in the case of our choice.

For example:

"Rocket Man"

Instead of going somewhere from the spaceport when the player carries something:
    let N be "[is-are the list of things carried by the player] really suitable gear to take to the moon?" in sentence case;
    say "[N][paragraph break]".

The Spaceport is a room. North of the Spaceport is the Rocket Launch Pad. The player carries a stuffed bear, a chocolate cookie, and a book.

The description of the book is "It is entitled [italic type]Why Not To Take [sentence cased inventory] To The Moon[roman type]."

To say sentence cased inventory:
    let N be "[a list of things carried by the player]" in title case;
    say "[N]".

Test me with "n / x book".

ExampleBlackout
Filtering the names of rooms printed while in darkness.

In this example, we want the names of rooms to be asterisked out if the player wanders around without the benefit of a candle. We can do this by

treating the room names as text, then replacing every letter:

```
"Blackout"

Tiny Room is a dark room. Absurdly Long-Named Room
is a dark room. It is west of Tiny Room.

The Candle Factory is north of Tiny Room. It contains a
beeswax candle. The beeswax candle is lit.

Rule for printing the name of a dark room:
    let N be "[location]";
    replace the regular expression "\w" in N with "*";
    say "[N]".

Test me with "w / look / e / n / get candle / s / w".
```

Notice that the hyphen in the Absurdly Long-Named Room does not get replaced. We could replace even that, if we liked, with

```
    replace the regular expression "\S" in N with "*";
```

which would catch every character that is not a space.

ExampleCurare

A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

```
"Curare"
```

```
A thing has a number called the last use. The last use of
a thing is usually 0.


Definition: a thing is old if its last use is 12 or less.
```

The actual number chosen in this definition is pretty much irrelevant: the main thing is that we want to establish relative values. The lower the "last use" number of an item, the older that item should be understood to be, as we see here:

```
To decide which thing is cyclically random (collection - a
description of objects):
    let choice be the oldest member of the collection;
    now the last use of the choice is the turn count;
    decide on choice.
```

This phrase will select, from the collection of objects passed to it, the one that has been mentioned least recently. This means that if we consult it repeatedly about the same collection, it will begin to cycle predictably; but if new items are added to the collection, it will mention these first before returning to the previous cycle. Now we can use this:

```
After taking inventory:
    say "You stare morosely at [the cyclically random
thing carried by the player], wondering what you're ever
going to find to do with it."
```

We could have said "You stare morosely at [the oldest thing carried by the player]" here, but doing so would not have set the "last use" property correctly, so we would not get the cycling behavior that we're looking for.

```
The Evidence Room is a room. Some shelves are
scenery supporters in the Evidence Room. A box is a
kind of container which is open and not openable. On
the shelves is a box. It contains a deformed bullet and a
driver's license.
```

```
The player carries a steel fish hook, a Chinese passport,
a tube of synthetic curare, and an envelope full of
Euros.


Test me with "i / i / i / i / i / i / i / get all from box / i / i / i".
```

Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

Suppose we are writing a game in which the mood of the piece changes, and we would like to have lots of descriptions that vary according to its current state. We might in that case want to create our own "by atmosphere" token, to control text variations, like this:

```
"Blink"


Atmosphere is a kind of value. The atmospheres are
normal, melancholy, and creepy.


The current atmosphere is an atmosphere that varies.


To say by atmosphere -- ending say_one_of with
marker I7_SOO_ATM:
    (- {-close-brace} -).
```

Since we're operating within the untyped Inform 6, we can make use of the fact that kinds of value are (internally) just constants, enumerated in the same order in which they were originally defined. In other words, "normal" at the I6 level translates to 1, "melancholy" to 2, and "creepy" to 3; so we can return the value of the current atmosphere, and thereby select option 1, 2, or 3:

```
Include (-
[ I7_SOO_ATM oldval count;
    if (count < (+ current atmosphere +)) return count;
    return (+ current atmosphere +); ];
-)
```

And that concludes the hard part. Now to test that it works:

```
The Flat is a room. "A small [one of]but
cozy[or]depressing[or]imprisoning[by atmosphere] flat.
Outside the window, the sun is [one of][or][or]apparently
[by atmosphere]shining and there is a brisk breeze
through the leaves of the birch trees. [one of]It would be
quite nice weather for a walk[or]The rest of the world
apparently has no appreciation of what you suffer[or]It
all looks deceptively normal[by atmosphere]."


Instead of waiting when the current atmosphere is
normal:
    say "Everything stretches wide and flat for just a
moment, as though all the world around you were
painted on a thin rubber sheet that is being [italic
type]stretched[roman type]. Then it snaps back into
place, leaving your ears ringing. But that little glitch was
enough to warn you. Someone is tampering with space-
time again. Someone very close by.";
    now the current atmosphere is creepy.


Test me with "look / z / look".
```

ExampleFun with Participles
Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on Clark's idle activity.

Mostly the Standard Rules use verbs adapted to finite forms ("he jumped", "we take the hammer", and so on). But Inform can also produce

participles to describe actions that are ongoing: "he is carrying the fedora" or "taking the hammer..."

In this example, we give non-player characters actions to perform and then have Inform dynamically describe what they're doing when the player chooses to look.

We start by establishing the idea that a verb can describe a particular action:

```
"Fun with Participles"


Section 1 - Descriptive Functionality


Describing relates various verbs to various action
names. The verb to describe means the describing
relation.


To look around is a verb. The verb look around
describes the looking action.


To stand about is a verb. The verb stand about
describes the waiting action. To look bored is a verb.
The verb look bored describes the waiting action. To
waste time is a verb. The verb waste time describes the
waiting action.


To jump is a verb. To leap is a verb. To pirouette is a
verb. The verb jump describes the jumping action. The
verb leap describes the jumping action. The verb
pirouette describes the jumping action.
```

Now we need to give every character some sort of idle activity. By default, we'll have people just be waiting, but allow for that idle activity to change into something more interesting if the player has told them to do something else.

```
A person has an action name called the current idle.
The current idle of a person is usually the waiting action.


Rule for writing a paragraph about someone (called
chosen person) when a verb describes the current idle
of the chosen person:
    say "[The chosen person] [are] here, [present
participle of a random verb that describes (the current
idle of the chosen person)]."


Instead of someone doing something:
    now the current idle of the person asked is (the action
name part of the current action);
    continue the action.


A persuasion rule:
    persuasion succeeds.


Section 2 - Scenario


Lab is a room. The fedora is a wearable thing in the
Lab. Clark is a man in the Lab.
```

And just to give past participles a test-drive as well, let's make Clark a bit of a drama king:

```
After Clark doing something when a verb describes (the
action name part of the current action):
    say "'Fine, have it your way!' Clark exclaims. 'But I
have [past participle of a random verb that describes
(the action name part of the current action)] for the last
time!'";
    rule succeeds.
```

Test me with "look / Clark, jump / look / Clark, look / look / Clark, wait".

ExampleVariety *WI*

Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

Verbs can be related to other things by relations. We've seen that it's possible for a verb to "mean" a relationship. But we can also create a relation between verbs and actions. For instance, we can tell Inform that "take", "get", and "acquire" are all valid ways to describe the action of taking, and then allow it to pick a verb randomly to describe whatever action just occurred.

"Variety"

Section 1 - Descriptive Functionality

Describing relates various verbs to various action names. The verb to describe means the describing relation.

To take is a verb. To acquire is a verb. To get is a verb.

The verb take describes the taking action. The verb acquire describes the taking action. The verb get describes the taking action.

To drop is a verb. To put down is a verb. To discard is a verb. The verb drop describes the dropping action. The verb put down describes the dropping action. The verb discard describes the dropping action.

To sniff is a verb. To smell is a verb. The verb sniff describes the smelling action. The verb smell describes the smelling action.

To jump is a verb. To leap is a verb. To pirouette is a verb. The verb jump describes the jumping action. The verb leap describes the jumping action. The verb pirouette describes the jumping action.

After an actor doing something when the noun is nothing and a verb describes (the action name part of the current action) (this is the apply random verbs to describing nounless actions rule):
    say "[The actor] [verb rendering applied to a random verb that describes (the action name part of the current action)].";
    rule succeeds.

After an actor doing something to something when a verb describes (the action name part of the current action) (this is the apply random verbs to describing actions rule):
    say "[The actor] [verb rendering applied to a random verb that describes (the action name part of the current action)] [the noun].";
    rule succeeds.

To decide which text is the rendering of (V - verb) (this is verb rendering):
    decide on "[adapt V]".

```
Section 2 - Scenario

Lab is a room. The table is here. The bat and the ball
are on the table.

Test me with "get ball / drop ball / get bat / drop bat /
smell ball".
```

ExampleVariety 2

This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

Some of our default actions establish relations between items in the world, and reporting on the relation ("You are now carrying the fedora") can be a valid response alongside reporting on the action itself ("You take the fedora").

To do this, we need to teach Inform explicitly which relations are the results of actions, then check this when reporting on actions:

```
"Variety 2"

Section 1 - Descriptive Functionality

Describing relates various verbs to various action
names. The verb to describe means the describing
relation.

Table of Action Results

relation action
```

the taking action
the wearing action
the taking off action

To take is a verb. To acquire is a verb. To get is a verb.

The verb take describes the taking action. The verb acquire describes the taking action. The verb get describes the taking action.

To drop is a verb. To put down is a verb. To discard is a verb. The verb drop describes the dropping action. The verb put down describes the dropping action. The verb discard describes the dropping action.

To sniff is a verb. To smell is a verb. The verb sniff describes the smelling action. The verb smell describes the smelling action.

To jump is a verb. To leap is a verb. To pirouette is a verb. The verb jump describes the jumping action. The verb leap describes the jumping action. The verb pirouette describes the jumping action.

To don is a verb. The verb don describes the wearing action.

To doff is a verb. The verb doff describes the taking off action.

```
After an actor doing something when the noun is
nothing and a verb describes (the action name part of
the current action) (this is the apply random verbs to
describing nounless actions rule):
    say "[The actor] [verb rendering applied to a random
verb that describes (the action name part of the current
action)].";
    rule succeeds.


After an actor doing something to something when a
verb describes (the action name part of the current
action) (this is the apply random verbs to describing
actions rule):
    let current action name be the action name part of the
current action;
    if a random chance of 1 in 2 succeeds and the current
action name is a related action listed in the Table of
Action Results:
        choose a row with the related action of current
action name in the Table of Action Results;
        let R be the relation entry;
        let subject be the actor;
        let chosen object be the noun;
        say "[The subject] [are] now [present participle of a
random verb that means R] [the chosen object].";
    else:
        say "[The actor] [verb rendering applied to a
random verb that describes (the action name part of the
current action)] [the noun].";
    rule succeeds.


To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".


To say infinitive of (V - a verb): (- {V}(1); -).


To say past participle of (V - a verb): (- {V}(2); -).
```

To say present participle of (V - a verb): (- {V}(3); -).

Section 2 - Scenario

Lab is a room. The fedora is a wearable thing in the Lab.

Test me with "wear the fedora / take off the fedora / wear fedora / take off fedora".

ExampleHistory Lab

We create phrases such as "the box we took" and "the newspaper Clark looked at" based on what has already happened in the story.

The examples Variety and Narrative Register show how verbs can be associated with particular actions. Here, we use the same principle so that we can report to the player what was last done to a particular object, either by the player or by someone else.

To do this, we need to use the idea of stored actions from the Advanced Actions chapter.

"History Lab"

Section 1 - Procedure

An object has an action called the last action.

Describing relates various verbs to various action

names. The verb to describe means the describing relation.

To take is a verb. The verb take describes the taking action.
To drop is a verb. The verb drop describes the dropping action.
To look at is a verb. The verb look at describes the examining action.
To examine is a verb. The verb examine describes the examining action.

After an actor doing something to something:
    if a verb describes the action name part of the current action:
        now the indefinite article of the noun is "the";
        now the last action of the noun is the current action;
    continue the action.

After printing the name of something (called item):
    if the last action of the item is not waiting and the last action of the item is not the current action:
        let chosen action-name be the action name part of the last action of the item;
        let chosen actor be the actor part of the the last action of the item;
        if a verb describes the chosen action-name:
            let the chosen verb be a random verb that describes the chosen action-name;
            say " [if the chosen actor is the player][we][else][chosen actor][end if] [adapt chosen verb in past tense]";

Section 2 - Scenario

Lab is a room. It contains a box. The box contains a newspaper. Clark is a man in the Lab.

```
A persuasion rule:
    persuasion succeeds.


Test me with "x box / look / x newspaper / look / clark, x
newspaper / clark, get box / clark, drop box / look / take
box / i / smell box / i".
```

Notice that smelling the box does not change the box's description because we haven't gotten around to defining a smell or sniff verb.

ExampleRelevant Relations

An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

Suppose that we wanted authors to be able to indicate which relations should or should not be included in room descriptions, and have the system dynamically honor that instruction.

Inform already knows about verbs for describing supporting, containment, carrying, and wearing, so we could write a set of instructions to handle such cases. To do this, we're using the "writing a paragraph about" activity, which is described in the chapter on activities.

```
"Relevant Relations"


Section 1 - Procedure


Rule for writing a paragraph about something (called
item):
    now the current paragraph is { };
    say "[one of][regarding item]There [are] [an item]
here[or][We] [can see] [an item] here[at random]. [run
paragraph on]";
    follow the descriptive rules for the item;
    repeat with new item running through the current
```

```
paragraph:
        now the prior named object is nothing;
        if new item is not the item:
            follow the descriptive rules for the new item;
    say paragraph break.


Rule for writing a paragraph about someone (called
chosen person):
    now the current paragraph is { };
    say "[one of][regarding chosen person][The chosen
person] [are] here[or][We] [can see] [a chosen person]
here[at random]. [run paragraph on]";
    follow the descriptive rules for the chosen person;
    repeat with new item running through the current
paragraph:
        now the prior named object is nothing;
        if new item is not the chosen person:
            follow the descriptive rules for the new item;
    say paragraph break.


The descriptive rules are an object-based rulebook.


Definition: a container is see-through:
    if it is transparent:
        yes;
    if it is open:
        yes;
    no.


A descriptive rule for a see-through container (called
item) (this is the describe contents rule):
    describe the containment relation for item.


A descriptive rule for a supporter (called item):
    describe the support relation for item.
```

```
A descriptive rule for a person (called item):
    describe the wearing relation for the item.

A descriptive rule for a person (called item):
    describe the carrying relation for the item.

The current paragraph is a list of things that varies.

Before printing the name of something (called
mentioned target) while writing a paragraph about
something:
    add the mentioned target to the current paragraph, if
absent.

To describe (R - a relation of objects) for (item - a thing):
    if a thing to which item relates by R is a thing:
        say "[The item with pronoun] [verb rendering
applied to a random verb that means R] [the list of
things to which item relates by R with indefinite articles].
[run paragraph on]"

To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".

To say (T - a thing) with pronoun:
    if T is the prior named object:
        say "[regarding T][They]";
    else:
        say "[The T]"

Section 2 - Scenario
```

```
The Space Elevator is a room. "Mercifully, there aren't
any windows. The ability to see how far up you are
would almost certainly make you ill."


The luggage rack is a supporter in the Space Elevator.
The suitcase is a closed openable container on the
luggage rack. The bouquet is on the luggage rack.


Clark is a man in the Space Elevator. Clark is carrying a
box of cupcakes. Clark is wearing a t-shirt. The
description of the box of cupcakes is "They're the latest
confection from Red Velvet Planet, the Martian bakery."


Persuasion rule: persuasion succeeds.
```

We can if we like then add alternate names for these relations that will be randomly swapped in some of the time. For instance:

```
To sport is a verb. The verb to sport means the wearing
relation.


To hold up is a verb. The verb to hold up means the
support relation.


Test me with "clark, drop the box / look / clark, take the
suitcase / look / clark, get bouquet".
```

One might, hypothetically, imagine going even further than this and simply designating relations as either "important" or "unimportant" -- perhaps changing the relation's designation at runtime. Relations are not themselves allowed to have properties, however.

A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

Sometimes it is more sensible to describe numbers roughly than in exact terms. For instance, we might want to have our player perceive "many people" rather than "forty-two people" on entering a room. To achieve this, we might write our own "to say" phrase.

```
"Ballpark"


To say (count - a number) in round numbers:
    repeat through the Table of Numerical Approximation:
        if count is less than threshold entry:
            say "[approximation entry]";
            rule succeeds.
```

Phrases will be explained more thoroughly in a later chapter, but as we have already seen in the examples, we can make a "To say..." phrase that will allow us to create our own text substitutions. In this case, we are going to replace the specific number with a vaguer one chosen from a chart, so:

```
Table of Numerical Approximation

threshold  approximation
1          "no"
2          "one"
3          "a couple of"
6          "a few"
13         "some"
21         "many"
100        "lots and lots of"
```

The idea here is that we will work our way through the table until we hit a line where the threshold number is higher than the number we want to express, and then print that output: so if we have less than one item, we'll print "no"; if we have more than none but less than two, we'll print "one"; if we have less than three, we'll print "a couple of"; if we have three, four, or five (but not six), we'll print "a few."

A room has a number called the population. The population of a room is usually 0. The description of a room is usually "You observe [population of the location in round numbers] [if population of the location is 1]person [otherwise]people [end if]here.".

The Stadium is a room. The Hot Dog Stand is west of the Stadium. The Women's Restroom is south of the Stadium.

The population of the Stadium is 500. The population of the Hot Dog Stand is 3. The population of the Restroom is 750.

Test me with "w / e / s".

Example **Fifty Times Fifty Ways**

Writing your own rules for how to carry out substitutions.

There is only so much we can cram into a text property, so being able to swap in properties is useful but limited. Fortunately, we can also, if we want, create new phrases for how to say things in brackets:

"Fifty Times Fifty Ways"

The Beekeeper's Palace is a room. Wasp is a woman in the palace. Drone is a man in the palace.

```
A person can be fierce or mellow. Wasp is fierce. Drone
is mellow. A person can be calm or angry. A person is
usually calm. A person has some text called insult. The
insult of a person is usually "Grasshopper". The insult of
Wasp is "Larva".



Instead of kissing someone:
    say "'[denial for the noun], [insult for the noun]!
[boast]!'";
```

Now to provide some meaning to these bracketed forms. We'll start with the easy one:

```
To say boast:
    say "I have ferocious allies".
```

This is a "to say" phrase; we will learn more about phrases in a later chapter, but for now it may be enough to observe that whatever we write after "to say..." becomes a valid substitution in bracketed speech. In this particular case there is no advantage to using the boast token rather than spelling the text out in the quotation, but we might in theory add further instructions to randomize the output, for instance.

To say phrases can be more complex, as well, since we can have them incorporate extra information:

```
To say insult for (speaker - a person):
    if speaker is angry, say "[the insult of the noun]";
    otherwise say "small one".
```

Here where we have (speaker - a person), we are leaving a slot which we can later fill in, madlibs-like, with any person we like. That is why we can write "insult for the noun": we are summoning the To say phrase and telling it to fill in the identity of the unknown speaker with the noun.

This differs from "insult of the noun" in the previous example; in that case, each person had his own insult property, and were merely printing that property out. Here we are actually telling Inform to calculate anew what the insult should be, and giving it some instructions about how to do that.

Our instructions can also get arbitrarily complex:

```
To say denial for (speaker - a person):
    if speaker is calm:
        say "You must not";
    otherwise if speaker is female:
        say "Stand back";
    otherwise:
        say "You forget yourself".


Instead of attacking someone:
    now the noun is angry;
    say "'Get away, [insult]!'"


Test me with "kiss wasp / hit wasp / kiss wasp / kiss
drone / hit drone / kiss drone".
```

So the effects we can get with text substitutions are quite flexible. We could even, if we wanted, fill in the substitutions by random choice, or by selecting items from a long list or table, should we have so bellicose a set of characters that they cannot make do with one or two insulting remarks apiece.

Example **Narrative Register** *WI*

248 Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

As we saw in "Variety", we can associate verbs with particular actions and call them up as needed. If we do that, though, we can also store additional information about those verbs and use that information to select the ideal verb to use in a particular situation.

In this example, we create a table of verbs and their meanings, together with some connotative information. Each time we report an action, we then score all the available verbs to decide which is the most suitable to use at the moment. This allows us to change the narrator's diction change mid-game and have the action descriptions change as well.

Moreover, because we're using adaptive verbs, these responses will automatically inflect properly even if we change the story tense and

viewpoint.

"Narrative Register"

Section 1 - Descriptive Functionality

Describing relates various verbs to various action names. The verb to describe means the describing relation.

To take is a verb. To acquire is a verb. To get is a verb. To gain is a verb. To obtain is a verb. To pick up is a verb. To bag is a verb. To procure is a verb. To score is a verb. To grab is a verb. To snag is a verb. To snatch is a verb. To collect is a verb.

To drop is a verb. To put down is a verb. To discard is a verb. To throw away is a verb. To dispose of is a verb. To set down is a verb. To toss aside is a verb. To ditch is a verb. To abandon is a verb. To dump is a verb. To jettison is a verb. To abjure is a verb. To foresake is a verb. To dispense with is a verb.

After an actor doing something to something when a verb describes (the action name part of the current action) (this is the apply random verbs to describing actions rule):
    score the relevant verbs;
    sort the Table of Verb Meanings in reverse relevance order;
    choose row 1 in the Table of Verb Meanings;
    let top score be the relevance entry;
    sort Table of Verb Meanings in random order;
    repeat through the Table of Verb Meanings:
        if relevance entry is top score:
            say "[The actor] [verb rendering applied to (word entry)] [the noun].";

```
            erase relevance;
            rule succeeds.


To decide which text is the rendering of (V - verb) (this
is verb rendering):
    decide on "[adapt V]".


To score the relevant verbs:
    repeat through the Table of Verb Meanings:
        if the meaning entry is (the action name part of the
current action):
            increase relevance entry by 1;
            repeat with chosen connotation running through
connotations entry:
                if the chosen connotation is listed in the
current register:
                    increase relevance entry by 1;
                otherwise:
                    decrease relevance entry by 1.


To erase relevance:
    repeat through Table of Verb Meanings:
        now relevance entry is 0.


A tonality is a kind of value. The tonalities are pompous,
archaic, slangy, upbeat, downbeat.


Connoting relates various verbs to various tonalities.
The verb to connote means the connoting relation.


The current register is a list of tonalities that varies. The
current register is { }.


When play begins:
```

    repeat through the Table of Verb Meanings:
        now the word entry describes the meaning entry;
        now relevance entry is 0;
        repeat with chosen tone running through the connotations entry:
            now the word entry connotes the chosen tone.

Table of Verb Meanings

word	meaning	connotations	relevance (a number)
take	the taking action	{ }	
acquire	the taking action	{pompous}	
get	the taking action	{ }	
gain	the taking action	{ }	
obtain	the taking action	{pompous}	
pick up	the taking action	{ }	
bag	the taking action	{slangy}	
[illegible]	the taking action	{slangy, urban}	
[illegible]	the taking action	{archaic}	
grab	the taking action	{slangy}	
snag	the taking action	{slangy}	
snatch	the taking action	{slangy}	
collect	the taking action	{ }	
discard	the dropping action	{pompous}	
drop	the dropping action	{ }	
put down	the dropping action	{ }	
toss aside	the dropping action	{ }	
ditch	the dropping action	{slangy}	
throw away	the dropping action	{ }	
dispose of	the dropping action	{ }	
set down	the dropping action	{ }	
abandon	the dropping action	{downbeat}	
give up	the dropping action	{downbeat}	
abjure	the dropping action	{archaic}	
forsake	the dropping action	{archaic}	
jettison	the dropping action	{pompous}	
dispense with	the dropping action	{pompous}	

Section 2 - Changing Tone Mid-Game

Understand "new tone" as changing the tone. Changing the tone is an action out of world.

```
Carry out changing the tone:
    now the current register is { };
    if a random chance of 1 in 4 succeeds:
        say "Your narrator will now adopt an ordinary
tone.";
        rule succeeds;
    let rando be a random tonality;
    add rando to the current register, if absent;
    say "Your narrator will now be [rando]."



Section 3 - Scenario



Lab is a room. The table is here. The bat and the ball
are on the table.



Test me with "get ball / drop ball / get bat / drop bat /
new tone / get all / drop all / new tone / get all / drop all".
```

ExampleStraw Into Gold

Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or "man" -- depending on which the player last used -- until the first time the player refers to him as "Rumpelstiltskin".

```
"Straw Into Gold"



The Cell is a room. Rumpelstiltskin is an improper-
named man in the Cell. Rumpelstiltskin can be identified
or unidentified. Rumpelstiltskin is unidentified.
```

```
R-name is a kind of value. The R-names are dwarf, guy,
dude, and man-thing. Rumpelstiltskin has an R-name.
Understand "[R-name]" as Rumpelstiltskin.
```

Our example is slightly complicated by the fact that "man" is a name already known to Inform, so we can't re-use it as a kind of value. This is possible to work around, though:

```
Understand "man" as man-thing.
```

Now we borrow from the Activities chapter to look at the exact wording of the player's command:

```
After reading a command:
    if the player's command includes "[R-name]",
        now the R-name of Rumpelstiltskin is the R-name
understood;
    if the player's command includes "Rumpelstiltskin":
        now Rumpelstiltskin is identified;
        now Rumpelstiltskin is proper-named.


Rule for printing the name of Rumpelstiltskin when
Rumpelstiltskin is unidentified:
    if the R-name of Rumpelstiltskin is man-thing:
        say "man";
    otherwise:
        say "[R-name]".


Test me with "x dwarf / x guy / x dude / look / x
rumpelstiltskin / look / x man".
```

Example First Name Basis
Allowing the player to use different synonyms to refer to something.

Sometimes we create objects that we want the player to be able to call by different names: a television that should also answer to "tv" and "telly",

for instance, or a refrigerator the player might also call "fridge". In this case, we can use instructions like

```
Understand "tv" and "telly" as the television.
```

to add extra names to the object we've defined.

```
"First Name Basis"


The Crew Lounge is a room. "Deliberately spartan: the
crew feels weight restrictions here first, so there aren't
any chairs, just a few thin pads on the ground."


The holographic projector is a device in the Crew
Lounge. "The one major source of entertainment is the
holographic projector, a top of the line Misthon 9000, on
which you view every beam you can get." Understand
"holo" or "holograph" or "Misthon" or "9000" as the
projector.


The description of the projector is "[if switched on]The
projector is now playing a documentary about the early
politics of the Mars colony.[otherwise]The air above the
projector is disappointingly clear.[end if]".
```

(This description is for local color; we will learn more about devices, and conditions like "if switched on", later in this chapter.)

By default, Inform does not understand the names of an object's kind as referring to that object, unless the object has no other name of its own. We can change this, if we like, by defining names that should be applied to everything of a given kind:

```
Lewis and Harper are men in the Crew Lounge.
Understand "man" or "guy" or "chap" or "lad" or "male"
as a man. Understand "men" or "chaps" or "lads" or
"guys" or "males" as the plural of a man.
```

```
The description of Lewis is "A wiry, excitable engineer
who just signed aboard last week." The description of
Harper is "Harper's a good guy: taciturn when sober,
affectionate when drunk, but rarely annoying in either
state."


Test me with "x holo / x man / lewis / x guy / harper /
turn on projector / x holo projector / get men".
```

Inform's naming abilities go considerably further, in fact: we can also instruct it to understand words only under certain circumstances, or only when they appear with other words. Fuller details may be found in the chapter on Understanding.

---

323 ExampleQuiz Show *WI*

In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

```
"Quiz Show" by Mike Tarbert


Use scoring.


Answer mode is a truth state that varies.
Current state is a text that varies.


Guessing is an action applying to one topic.
Understand "[text]" as guessing when answer mode is
true.
```

Because of the "...when" part of this line, random text is only treated as

an answer when a question is being asked.

```
Check guessing (this is the default wrong answer rule):
    if the topic understood is not a topic listed in the Table
of Dates of Statehood:
        say "Wrong!";
    now answer mode is false.
```

```
Carry out guessing a topic listed in the Table of Dates of
Statehood:
    if state entry is the current state:
        say "Correct! ([comment entry], to be exact!)";
        increase the score by one;
    otherwise:
        say "Wrong!";
    now answer mode is false.
```

This next rule allows a player to do something other than answer the question, but then makes him wait for another question before answering.

```
Before doing anything other than guessing:
    if answer mode is true:
        say "(ignoring the question)[line break]";
    now answer mode is false.
```

```
Section 2 - Scenario
```

```
The Lab is a room. Sam is a man in the lab.
```

```
Every turn when the player is in the lab:
    if a random chance of 3 in 5 succeeds:
        choose a random row in the Table of Dates of
Statehood;
        say "Sam asks you, 'In what year was [state entry]
admitted into the Union?'";
        now current state is state entry;
```

now answer mode is true.

Table of Dates of Statehood

Topic Statement
"Delaware 7th"

Test me with "1845 / z / z / 1787 / 1792 / z / 1845 / g".

Note that the situation will become a little more complicated if we have two or more identical topics in our trivia list; in that case, we would need to loop through the Table of Dates of Statehood explicitly, and only mark the player wrong if none of the lines were found to match. (See the chapter on Tables for many more ways to manipulate table behavior.)

Example **Pot of Petunias**

Responding sensibly to a pot of petunias falling from the sky.

Suppose we have an object that makes a dramatic entrance on the scene, like so:

"Pot of Petunias"

Wide Open Field is a room. "A big field under a big sky. The clouds are puffy, the trees are handsome."

Some clouds and some trees are scenery in Wide Open Field. The description of the clouds is "That one looks like Yoda's head." The description of the trees is "You've never been much good at botany, so it's

anyone's guess what kind they are."

A rock is in Wide Open Field. The description of the rock is "It looks like it's been here from the dawn of time."

The broken flower pot is a thing. The description of the broken flower pot is "It contains the remains of some abused petunias."

At 9:01 am:
    move the broken flower pot to the location;
    say "Quite unexpectedly, a flower pot falls from the sky and breaks open on the ground. Good thing you weren't standing six inches to the left.";
    set pronouns from the broken flower pot.

Test me with "x it / x it / x it".

If we leave out the "set pronouns..." line here, we'll wind up with the following very unsatisfactory end to our test transcript:

Quite unexpectedly, a flower pot falls from the sky and breaks open on the ground. Good thing you weren't standing six inches to the left.

>[3] x it
It looks like it's been here from the dawn of time.

Occasionally it is useful to give something a printed name because we want to call it something extremely long-winded; give one thing a name that is the subset of the name of something else; or use words such as "with" or "and" that are likely to confuse Inform into thinking that the object name ends before it actually does.

Often it is enough to preface these ambiguously-titled things with "a thing called..." or "a supporter called..." or the like, as here:

```
South of Spring Rolls is a room called Hot and Sour
Soup.
```

prevents Inform from trying to read "Hot and Sour Soup" as two separate rooms, while

```
The player carries an orange ticket. The player carries a
thing called an orange.
```

creates two objects instead of the one orange ticket that would result if the second sentence were merely "The player carries an orange."

Really long names can be a bit cumbersome. For example:

```
The player carries a thing called an incriminating
photograph of a woman with blonde hair.
```

So we might instead give the photograph a printed name:

```
"Laura"
```

```
The City of Angels is a room. The incriminating
photograph is carried by the player. The printed name of
the incriminating photograph is "incriminating
photograph of a woman with blonde hair".
```

Now we've gotten around any awkwardness with printing the name -- but we also need to understand when the player refers to the photograph. When we define the names of objects under normal circumstances, Inform takes care of this automatically, but if we have especially set the

printed name, we must also specially define the appropriate terms for the player to use. For this we need "understand", which will be explained in much more depth in a later chapter:

```
Understand "woman" or "with" or "blonde" or "hair" or
"of" or "a" as the incriminating photograph.


Test one with "x photograph / x incriminating
photograph of a woman with blonde hair / x hair / x
blonde / x woman with blonde hair / x incriminating
photograph of a woman".
```

That's probably as far as we really need to go, and if you are satisfied with this behavior, there is no need to read on.

One possible objection to this solution is that Inform will accept some nonsensical formulations as applying to the photograph: for instance, it will allow >EXAMINE PHOTOGRAPH OF, >X BLONDE PHOTOGRAPH WOMAN INCRIMINATING, or even >X OF ...though in the case there were two items with "of" names, the game would disambiguate with a question such as "Which do you mean, the incriminating photograph of a woman with blonde hair or the essence of wormwood?"

Traditionally, Inform has tended to be fairly flexible about word order, preferring to err in the direction of leniency. On the other hand, there are times when we need more exacting rules in order to distinguish otherwise similar cases.

Two features allow us to specify more exactly if we so desire. The first is that, if we specify a whole phrase as the name of something, all the words in that phrase are required, in the order given. Thus "Understand "blonde hair" as the photograph" would require that both "blonde" and "hair" be present, and would not recognize >X BLONDE, >X HAIR BLONDE, or >X HAIR.

Second, we can create tokens, such as "Understand "blonde hair" or "hair" as "[hair]", and then use these tokens in match phrases. This saves a good deal of time when we want to specify a number of different but fussy alternatives. So, for instance, here is a drawing that would not respond to >X OF, or >X BROWN EYES, but would respond to >X DRAWING OF MAN WITH BROWN EYES, >X MAN WITH BROWN EYES, and so on:

```
The drawing is carried by the player. The printed name
of the drawing is "drawing of a man with brown eyes".


Understand "eyes" or "brown eyes" as "[brown eyes]".
Understand "man" or "man with [brown eyes]" or
"brown-eyed man" as "[man]". Understand "[man]" or
"drawing of [man]" or "drawing of a [man]" as the
drawing.


Test me with "test one / test two".


Test two with "x drawing / x man / x of / x drawing of
man / x drawing of a man / x drawing of a man with
brown eyes / x drawing of a brown-eyed man / x brown
eyes".
```

Further refinements are possible: the "privately-named" attribute tells Inform not to try to understand the source name of an object at all, so if we write

```
The purple rabbit is a privately-named thing.
```

...the player will not be able to refer to it as "purple" or "rabbit" or "purple rabbit".

There are also ways to make names to refer to entire kinds of objects (so "dude" will refer to any man in the game); to specify names that only refer to objects in the plural (so GET PICTURES will pick up several pictures together); to reflect an object's properties (so "red apple" works only as long as the apple is in fact red); or even to refer to the object's relationships to other objects (so "bottle of wine" works only when wine is indeed in the bottle). All these refinements are discussed in the chapter on Understanding.

The Understanding chapter lays out ways to change how the player can refer to objects, but we may not want to wait that long for some of the basic features. Here, for instance, is how to add synonyms that the player can use to refer to an entire kind of object:

```
"Vouvray"

The Wine Emporium is a room. "Set aside, you rather
suspect, for tourists: this chamber is barrel-vaulted
stone, lined on each side with casks of aging wine.
Discarded brochures here and there advertise Wine
Tours of the Loire Valley in three different languages,
none of them French."

A cask is a kind of thing. A cask is always fixed in place.
Understand "cask" or "barrel" as a cask. Understand
"casks" or "barrels" as the plural of cask.

The Vouvray cask and the Muscadet cask are casks in
the Wine Emporium.

Test me with "get barrels / get barrel / muscadet / x
casks / x muscadet cask".
```

ExampleNorth by Northwest

Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed words.

Suppose we wanted to add intermediate compass directions such as north-northwest to our game. Because of the limitations of the index map, we won't be able to view these connections on the world map, but we can

certainly create them, and use them in route-finding, just like other directions.

Here's how we'd set up such a thing:

"North by Northwest"

Section 1 - Procedure

The north-northwest is a direction. North-northwest has opposite south-southeast. Understand "n-nw" or "nnw" as north-northwest.

The north-northeast is a direction. North-northeast has opposite south-southwest. Understand "n-ne" or "nne" as north-northeast.

The south-southwest is a direction. South-southwest has opposite north-northeast. Understand "s-sw" or "ssw" as north-northwest.

The south-southeast is a direction. South-southeast has opposite north-northwest. Understand "s-se" or "sse" as south-southeast.

The west-northwest is a direction. West-northwest has opposite east-southeast. Understand "w-nw" or "wnw" as west-northwest.

The east-northeast is a direction. East-northeast has opposite west-southwest. Understand "e-ne" or "ene" as east-northeast.

```
The west-southwest is a direction. West-southwest has
opposite east-northeast. Understand "w-sw" or "wsw" as
west-northwest.
```

```
The east-southeast is a direction. East-southeast has
opposite west-northwest. Understand "e-se" or "ese" as
east-southeast.
```

A complication arises because we reach the 9-character limit: Inform truncates the names of objects to nine characters before trying to understand them. To make matters worse, the hyphen (and other punctuation marks) count as two letters. So both north-northwest and north-northeast will get truncated to "north-no", and be indistinguishable when the player types them.

When we are compiling for Glulx, the limit is easily changed with a single line, setting the constant called DICT_WORD_SIZE. For instance, if we wanted to raise the limit to 15, we would simply write "Use DICT_WORD_SIZE of 15."

If we're compiling to the Z-machine, however, we'll have to resort to some manipulation of the player's command. The general solution is that when the player's name for an object is going to have to be longer than we can correctly read, we can substitute an unambiguous abbreviation for the thing the player typed. In this case, it will be simplest and most efficient always to condense the player's direction names to single letters, thus:

```
After reading a command:
    let N be "[the player's command]";
    replace the text "north" in N with "n";
    replace the text "east" in N with "e";
    replace the text "south" in N with "s";
    replace the text "west" in N with "w";
    change the text of the player's command to N.
```

For more on the use of text, see the Advanced Text chapter.

Section 2 - Scenario

```
The Empty Field is north-northwest of the Deserted
Road.


A crop-dusting plane is a backdrop. It is not scenery. It
is in the Deserted Road and Empty Field. The initial
appearance of the crop-dusting plane is "[one of]In the
distance[or]Approaching faster and faster[or]Flying
ominously low and directly towards you[or]Immediately
overhead[or]Circling around for another
approach[cycling] is a standard crop-dusting plane."


After looking:
    say "From here you can run to [the list of adjacent
rooms]."


Rule for printing the name of a room (called the target)
which is not the location while looking:
    let chosen direction be the best route from the
location to the target;
    say "[chosen direction]".


Test me with "sse / north-northwest".
```

In practice, this is going to be overkill for almost all games: most players already find eight compass directions plus up and down to be enough (or more than enough) to keep track of. But the option exists, in case there is a compelling reason to use it.

(Note also that we are allowed to use multi-word direction names, so we could have called the directions "north by northwest", "north by northeast", and so on. This example deliberately takes the hard way in order to show how to resolve the nine-character problem.)

Example **Igpay Atinlay**

A pig Latin filter for the player's commands.

For the sake of argument, suppose we want to parrot back all the player's commands in pig Latin:

```
"Igpay Atinlay"

Armfay is a room.

After reading a command:
    let N be "[the player's command]";
    replace the regular expression "\b(<aeiou>+)(\w*)" in
N with "\1\2ay";
    replace the regular expression
"\b(<bcdfghjklmnpqrstvwxz>+)(\w*)" in N with "\2\1ay";
    say "[N][paragraph break]";
    reject the player's command.

Test me with "nix on the stupid".
```

Letting the player guess types for an unidentifiable fish.

Suppose we have an unhappily mutated fish that the player can refer to by any of a number of species names, or any word followed by -fish. We want to reject these commands, but preserve a memory of what the player last tried to call the thing:

```
"Mr. Burns' Repast"
```

Wharf is a room.

There is an unknown fish in the Wharf. The unknown fish has some a text called the supposed name. The description of the unknown fish is "The victim of heavy mutagens, this thing is not really recognizable as any species you know.".

Fish variety is a kind of value. The fish varieties are salmon, albacore, mackerel.

Rule for printing the name of the unknown fish:
    if the supposed name of the unknown fish is "", say the printed name of the unknown fish;
    otherwise say the supposed name of the unknown fish.

After reading a command:
    if the unknown fish is visible and player's command matches the regular expression "\b\w+fish":
        let N be "[the player's command]";
        replace the regular expression ".*(?=\b\w+fish)" in N with "";
        now N is "[N](?)";
        now the supposed name of the unknown fish is N;
        respond with doubt;
        reject the player's command;
    otherwise if the unknown fish is visible and the player's command includes "[fish variety]":
        now supposed name of the fish is "[fish variety understood](?)";
        respond with doubt;
        reject the player's command.

To respond with doubt:
    say "You're not [italic type]sure[roman type] you're seeing any such thing."

Test me with "get swordfish / look / touch monkfish / look / listen to tunafish / x fish / x salmon / look".

Example **Terracottissima Maxima**

WI

Flowerpots with textual names that might change during play.

Inform can also understand text properties:

"Terracottissima Maxima"

A flowerpot is a kind of thing. A flowerpot has a text called pattern. Understand the pattern property as describing a flowerpot. The printed name of a flowerpot is usually "[pattern] flowerpot". The printed plural name of a flowerpot is usually "[pattern] flowerpots".

The Herb Garden is a room. In the Herb Garden is a flowerpot with pattern "blue willow". In the Herb Garden is a flowerpot with pattern "striped". In the Herb Garden is a flowerpot with pattern "striped".

Test me with "x blue willow / get striped / look".

This may not seem very much different from having the pattern be a kind of value -- except that texts can, of course, hold almost anything. Further exploration of these possibilities may be found in the chapter on Advanced Text.

Example **Xot**

In Hitchhiker's Guide to the Galaxy, any erroneous command the player types can return to haunt him later in the game. We could do the same, if we liked, by storing the player's command whenever we print a parser error.

```
"Xot"

Humiliation Chamber is a room. "A grim, grey-walled
room. Cameras watch you from every angle; convex
mirrors reflect your actions; and up near the ceiling,
where you can't disable it, is a loudspeaker."

The last error is a text that varies. The last error is "xot".

Before printing a parser error:
    now the last error is the player's command.

Every turn when a random chance of 1 in 2 succeeds:
    say "Over the loudspeaker comes some distorted
nonsense. If you listen carefully, it sounds as though
some fool is saying '[last error], [last error], [last error]!'"

Test me with "wiggle / z / z / z / z / z / z".
```

Chapter 3: Place

Continuous Spaces and The Outdoors; *§3.5. Doors, Staircases, and Bridges*; *§3.6. Windows*; *§3.7. Ligh*

Contents of *The Inform Recipe Book*
Chapter 2: Adaptive Prose
Chapter 4: Time and Plot
Indexes of the examples

§3.1. Room Descriptions

The printing of a room description is a more delicate business than it might initially seem to be: Inform has to consider all the objects that the player might have brought into the room or dropped there, and all the objects on visible supporters, and decide how to group and list them.

All of this behavior is handled by the looking command, so we find the relevant rules in the carry out looking rulebook. To go through the elements step by step:

Looking begins by printing the name and description of the room we're in. We can introduce variations into room names and descriptions by changing their printed name and description properties, as in

```
now the printed name of the Church is "Lightning-Struck Ruin";
now the description of the Church is "The beams overhead have
been burnt away and the pews are charred. Only the stone walls
remain.";
```

If we need more drastic effects, we can turn off or change either of these features by altering the rules in the carry out looking rulebook. For instance, to remove the name of the location entirely from room descriptions, we would write

```
The room description heading rule is not listed in the carry out
looking rules.
```

(A word of warning: there is one other context in which the story prints a room name — when restoring a save or undoing a move. To omit the room title here too, add

```
Rule for printing the name of a room: do nothing.)
```

Ant-Sensitive Sunglasses demonstrates how to use activities to make more flexible room description text.

Next, the story determines what items are visible to the player and need to be described. These never include the player himself, or scenery, but other things in the environment will be made "marked for listing". This is also the stage at which Inform chooses the order in which items will be listed.

We are allowed to meddle by changing the priorities of objects, in case we want some things to be described to the player first or last in the room description; **Priority Lab** goes into detail about how. We can also force things to be left out entirely: **Low Light** handles the case of an object that can only be seen when an extra lamp is switched on, even though the room is not otherwise considered dark. **Copper River** implements the idea of "interesting" and "dull" objects: the story determines which items are currently important to the puzzles or narrative and mentions those in the room description, while suppressing everything else.

Then Inform carries out the writing a paragraph about... activity with anything that provides one; anything it prints the name of, it tags "mentioned". Thus

```
Rule for writing a paragraph about Mr Wickham:
    say "Mr Wickham looks speculatively at [list of women in the
location]."
```

will count Wickham and everyone he looks at as all having been mentioned, and will not refer to them again through the rest of the room description. More complicated uses of writing a paragraph abound. A developed system for handling supporters that don't list contents appears in **The Eye of the Idol**.

Inform then prints the initial appearances of objects that are marked for listing but not already mentioned; and then it performs the listing nondescript items activity, collating the remaining objects into a paragraph like

```
You can see a dog, a hen, ...
```

We can pre-empt items from appearing in this paragraph or change their listing by intervening with a Before listing nondescript items... rule, as in

```
Before listing nondescript items when the player needs the watch:
```

```
if the watch is marked for listing:
    say "The watch catches your eye.";
    now the watch is not marked for listing.
```

If we wanted the watch always to be listed this way, it would be better to give it an initial appearance, but for conditional cases, the listing nondescript items activity is a good place to intervene. For instance, **Rip** uses this activity to incorporate changeable or portable items into the main description text for a room when (and only when) that is appropriate.

The listing nondescript items activity also allows us to replace the "You can see..." tag with something else more fitting, if for instance we are in a dimly lit room.

When the story compiles the list of nondescript items, it adds tags such as "(open)" or "(empty)" or "(on which is a fish tank)" to the names of containers and supporters. We can suppress or change the "(empty)" tag with the printing room description details of activity, as in

```
Rule for printing room description details: stop.
```

And we can suppress the "(open)" and "(on which is...)" sorts of tags with the "omit the contents in listing" phrase, as in

```
Rule for printing the name of the bottle while not inserting or
removing:
    if the bottle contains sand, say "bottle of sand";
    otherwise say "empty bottle";
    omit contents in listing.
```

Finally, the looking command lists visible non-scenery items that sit on scenery supporters, as in

```
On the table is a folded newspaper.
```

These paragraphs can be manipulated with the printing the locale description activity and the printing a locale paragraph about activity.

Another common thing we may want to do is change the description of a room depending on whether we've been there before (as in **Slightly Wrong**) or on how often we've visited (as in **Infiltration**). **Night Sky**, meanwhile, changes the description of a room when we've examined another object, so that the player's awareness of his environment is affected by other things the character knows.

See **Looking** *for ways to change the default length of room descriptions*

Start of Chapter 3: Place

Back to Chapter 2: Adaptive Prose: §2.3. Using the Player's Input

Onward to §3.2. Map

Example 147: **Night Sky** A room which changes its description depending on whether an object has been examined.

Example 152: **Infiltration** A room whose description changes depending on the number of times the player has visited.

Example 334: **Ant-Sensitive Sunglasses** What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

Example 355: **Rip Van Winkle** A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

Example 358: **Priority Lab** A debugging rule useful for checking the priorities of objects about to be listed.

Example 359: **Low Light** An object that is only visible and

manipulable when a bright light fixture is on.

Example 4: ★★ **Slightly Wrong** A room whose description changes slightly after our first visit there.

Example 357: ★★ **The Eye of the Idol** A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

Example 362: ★★★ **Copper River** Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

---

§3.2. Map

A work of IF contains many spectacles and activities, and these must not all present themselves at once, or the player will be overwhelmed. One way to spread them out is in time, by having them available only as a plot develops, but another is to spread them out literally in space. The player has to walk between the Library and the Swimming Pool, and thus bookish and athletic tasks are not both presenting themselves at once. There have been valiant "one-room" IFs, and it forms a respectable sub-genre of the art, but most works of any size need a map.

Inform, following IF conventions, divides the world up into locations called "rooms", connected together by so-called "map connections" along compass bearings. Thus:

```
The Library is east of the Swimming Pool.
```

The example **Port Royal 1** develops a medium-sized map from such sentences. This

develops in **Port Royal 2** to include connections which bend around, allowing the rooms not to lie on an imaginary square grid.

Because it is useful to group rooms together under names describing whole areas, Inform also allows rooms to be placed in "regions". Thus:

```
The Campus Area is a region. The Library and the Swimming Pool
are in the Campus Area.
```

Port Royal 3 demonstrates this further. **A&E** shows how regions can be used to write simple rules which regulate access to and from whole areas of the map.

Many old-school IF puzzles involve journeys through the map which are confused, randomised or otherwise frustrated: see **Bee Chambers** for a typical maze, **Zork II** for a randomised connection, **Prisoner's Dilemma** for a change in the map occurring during play. A completely random map takes us away from traditional IF and more towards a different sort of old-school game, the computerised role-playing game with its endless quests through dungeons with randomly generated treasures and monsters. This style of map - building itself one step at a time, as the player explores - can sometimes be useful to provide an illusion of infinite expanse: see **All Roads Lead To Mars**.

While the standard compass directions are conventional in IF, there are times when we may want to replace them without other forms of directional relationship. **Indirection** renames the compass directions to correspond to primary colors, as in Mayan thinking. **The World of Charles S. Roberts** substitutes new ones, instead, introducing a hex-grid map in place of the usual one.

See **Going, Pushing Things in Directions** *for ways to add more relative directions, such as context-sensitive understanding of OUT and IN*

See **Room Descriptions** *for ways to modify the room description printed*

See **Ships, Trains and Elevators** *for rooms which move around in the map and for directions aboard a ship*

Start of Chapter 3: Place

Back to §3.1. Room Descriptions

Onward to §3.3. Position Within Rooms

Example 5: **Port Royal 1** A partial implementation of Port Royal, Jamaica, set before the

earthquake of 1692 demolished large portions of the city.

Example 8: **Port Royal 2** Another part of Port Royal, with less typical map connections.

Example 10: **Port Royal 3** Division of Port Royal into regions.

Example 78: **All Roads Lead to Mars** Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

Example 125: **Bee Chambers** A maze with directions between rooms randomized at the start of play.

Example 134: **Zork II** A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby.

Example 286: **Indirection** Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

Example 40: **Prisoner's Dilemma** A button that causes a previously non-existent exit to come into being.

Example 41: **The World of Charles S. Roberts** Replacing the ordinary compass bearings with a set of six directions to

impose a hexagonal rather than square grid on the landscape.

Example 101: ★★ **A&E** Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

---

§3.3. Position Within Rooms

Inform's division of geography into "rooms" is a good compromise for most purposes. The rooms are cut off from each other by (imaginary or actual) walls, while all of the interior of a given room is regarded as the same place.

Suppose we want things to happen differently in different corners of the same room? Inform can already do this a little, in that the player can be inside an enterable container or on an enterable supporter. For instance:

```
Instead of opening a door when the player is on the bed, say "You
can't reach the handle from the bed."
```

If we need to have divided-up areas of the floor itself, the standard approach is to define a small number of named positions. We then need to remember at which of these locations the player (or something else) currently stands.

Further Reasons Why All Poets Are Liars allows the player to be in different parts of a room by standing on a box which can be in different places: thus only the box needs an internal position, not the player, simplifying matters neatly.

Another interesting case is when one room is entirely inside another (such as a hut in a field, or a booth in a large convention hall), so that the exterior of the room should be visible from another location. **Starry Void** gives a simple demonstration of a magician's booth that can be examined from the outside, opened and closed, and entered to reach a new location.

★ *See* **Continuous Spaces and The Outdoors** *for making the space between rooms continuous*

★ *See* **Combat and Death** *for the use of position in a room in determining combat maneuvers*

★ *See* **Entering and Exiting, Sitting and Standing** *for automatically getting up from chairs before going places*

See **The Human Body** *for letting the player take different postures on furniture or on the floor*

See **Furniture** *for cages, beds, and other kinds of enterable supporters and containers*

Start of Chapter 3: Place

Back to §3.2. Map

Onward to §3.4. Continuous Spaces and The Outdoors

Example 199: **Further Reasons Why All Poets Are Liars** The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

Example 7: **Starry Void** Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

---

§3.4. Continuous Spaces and The Outdoors

Suppose we want to blur the boundaries between rooms, in an environment where there are no walls: out of doors, for instance?

The simplest cases involve making something exceptional visible in more than one place. **Carnivale** features an exceptionally large landmark seen by day; **Eddystone** an exceptionally bright one by night. **Waterworld** allows a very distant object (the Sun) to be seen throughout many rooms, but never approached. **View of Green Hills** gives the player an explicit command for looking through into an adjacent room.

Three systematic examples then present outdoor landscapes with increasing sophistication. **Tiny Garden** gives the multiple rooms of an extended lawn descriptions which automatically adapt to say which directions lead into further lawn area. **Rock Garden** provides a relation, "connected with", between rooms, allowing items in one to be seen from the other: an attempt to interact with a visible item in a different area of the garden triggers an implicit going action first. **Stately Gardens** provides a much larger outdoor area, where larger landmarks are visible from further away, and room descriptions are highly adaptive.

In an outdoor environment, the distinction between a one-move journey and a multiple-move journey is also blurred. **Hotel Stechelberg** shows a signpost which treats these equally.

See **Position Within Rooms** *for making the space within a room continuous*

See **Windows** *for another way to see between locations*

See **Doors, Staircases, and Bridges** *for still a third way to be told at least what lies adjacent*

See **Passers-By, Weather and Astronomical Events** *for more on describing the sky*

Start of Chapter 3: Place

Back to §3.3. Position Within Rooms

Onward to §3.5. Doors, Staircases, and Bridges

Example 216: ★ **Waterworld** A backdrop which the player can examine, but cannot interact with in any other way.

Example 62: ★★ **Tiny Garden** A lawn made up of several rooms, with part of the description written automatically.

Example 79: ★★ **Hotel Stechelberg** Signposts such as those provided on hiking paths in the Swiss Alps, which show the correct direction and hiking time to all other locations.

Example 212: ★★ **Carnivale** An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

Example 213: ★★ **Eddystone**

Creating new commands involving the standard compass directions.

Example 366: ★★ **Rock Garden** A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

Example 80: ★★★ **A View of Green Hills** A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

Example 367: ★★★ **Stately Gardens** An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

---

§3.5. Doors, Staircases, and Bridges

Inform's "door" kind provides for a tangible thing which comes between one room and another. A door can be open or closed, and openable or not: it can be locked or unlocked, and lockable or not. Here we create a conventional door, a natural gap in the rocks, and a (fixed in place) wooden ladder:

```
The fire door is an open door. The fire door is east of the Projection
Booth and west of the Fire Escape.
The narrow crevice is an open unopenable door. The crevice is east
of the Col du Prafleuri and west of Rocky Knoll Above Arolla.
The wooden ladder is an open unopenable door. The ladder is above
the Stableyard and below the Hay Loft.
```

Most doors are visible from both sides: they are single objects but present in two rooms at once, which raises a number of complications. Inform normally uses the same description looking from each way, which is not very interesting: **When?** and **Whence?** demonstrate neat ways to describe the two sides differently, and **Whither?** adds the option for the player to refer to doors as "the west door" and "the east door" automatically.

Neighbourhood Watch goes further by making a door behave differently on each side: from the "outside" you need a key, but "inside" it opens on a latch. Finally, **Garibaldi 1** shows how to access information about the two sides of a door.

Higher Calling demonstrates doors which automatically open as needed: though using the Inform extension Locksmith by Emily Short is probably easier and better. **Elsie**, conversely, demonstrates a door that closes one turn after the player has opened it.

Certain complications apply when characters other than the player have to see and interact with doors that exist in other rooms. **Wainwright Acts** demonstrates the syntax needed to handle this technically quirky situation.

Something Narsty and **Hayseed** provide a "staircase" kind useful for vertically arranged, always-open doors like staircases and (fixed in place) ladders.

One Short Plank implements a precarious plank bridge across a chasm as an open unopenable door.

See **Windows** *for climbing through a window from one room to another*

See **Ropes** *for portable connections between rooms, much of the development of which could be adapted to handle portable ladders. "Doors" are never allowed to move*

See **Magic (Breaking the Laws of Physics)** *for a hat that lets the player walk through closed doors*

See **Modifying Existing Commands** *for ways to allow the player to unlock with a key he isn't currently holding*

Start of Chapter 3: Place

Back to §3.4. Continuous Spaces and The Outdoors

Onward to §3.6. Windows

Example 45: **Something Narsty** A staircase always open and never openable.

Example 63: ★ **When?** A door whose description says "...leads east" in one place and "...leads west" in the other.

Example 89: ★ **Hayseed** A refinement of our staircase kind which can be climbed.

Example 128: ★ **Higher Calling** All doors in the game automatically attempt to open if the player approaches them when they are closed.

Example 236: ★ **Wainwright Acts** A technical note about checking the location of door objects when characters other than the player are interacting with them.

Example 321: ★ **Whither?** A door whose description says where it leads; and which automatically understands references such as "the west door" and "the east door" depending on which direction it leads from the location.

Example 24: ★★ **Neighborhood Watch** A locked door that can be locked or unlocked without a key from one side, but not from the other.

Example 106: ★★ **One Short Plank** A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

Example 151: ★★ **Elsie** A door that closes automatically one turn after the player opens it.

Example 22: ★★★ **Garibaldi 1** Providing a security readout device by which the player can check on the status of all doors in the game.

Example 64: ★★★ **Whence?** A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

---

§3.6. Windows

Calvin Coolidge once described windows as "rectangles of glass." For us, they have two purposes: first, they offer a view of landscape beyond. In the simplest case the view is of an area which will not be interacted with in play, and therefore does not need to adapt to whatever may have changed there:

> The window is scenery in the Turret. "Through the window you see miles and miles of unbroken forest, turning from green to flame in the hard early autumn."

More interesting is to adapt the view a little to provide a changing picture: a forest may not change much, but a street scene will. **Port Royal 4** allows us to glimpse random passers-by.

The trickiest kind of window allows the player to see another room which can also be encountered in play, and to interact with what is there. **Dinner is Served** presents a shop window, allowing people to see inside from the street, and even to reach through.

Vitrine handles the complication of a window misting up to become opaque, and thus temporarily hiding its view.

Second, windows provide openings in walls and can act as conduits. **Escape** shows how a "door" in the Inform sense can become a window. **A Haughty Spirit** provides a general kind of window for jumping down out of: ideal for escapers from Colditz-like

castles.

See **Doors, Staircases, and Bridges** *for a door which can be partially seen through*

Start of Chapter 3: Place

Back to §3.5. Doors, Staircases, and Bridges

Onward to §3.7. Lighting

Example 116: **Vitrine** An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

Example 21: **Escape** Window that can be climbed through or looked through.

Example 217: **Dinner is Served** A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

Example 272: **Port Royal 4** A cell window through which the player can see people who were in Port Royal in the current year of game-time.

Example 181: **A Haughty Spirit** Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

---

§3.7. Lighting

At any place (room, or inside a container) light is either fully present or fully absent. Inform does not usually try to track intermediate states of lighting, but see **The Undertomb 2** for a single lantern with varying light levels and **Zorn of Zorna** for multiple candles that can be lit for cumulative changes to the light level.

Light can be added to, but not taken away: rooms and things can act as sources of light, by having the "lighted" and "lit" properties respectively, but they cannot be sinks which drain light away. The reason darkness is not a constant hazard in Inform-written games is that rooms always have the "lighted" property unless declared "dark". (We assume daylight or some always-on electric lighting.) A "dark" room may well still be illuminated if a light source happens to be present:

> The Deep Crypt is a dark room. The candle lantern is a lit thing in the Deep Crypt.

Hymenaeus allows us to explicitly refer to torches as "lit" or "unlit", or (as synonyms) "flaming" or "extinguished".

For light produced electrically we might want a wall switch, as in **Down Below**, or a portable lamp, as in **The Dark Ages Revisited**.

The fierce, locally confined light thrown out by a carried lamp has a quality quite unlike weak but ambient daylight, and **Reflections** exploits this to make a lantern feel more realistic.

When the player experiences darkness in a location, Inform is usually very guarded in what it reveals. ("It is pitch dark, and you can't see a thing.") **Hohmann Transfer** gives darkness a quite different look, and **Four Stars** heightens the other senses so that a player in darkness can still detect her surroundings. The first of the two examples in **Peeled** allows exploration of a dark place by touch.

It is sometimes useful to check whether a room that is not the current location happens to contain a light source or be naturally lighted. This poses a few challenges. **Unblinking** demonstrates one way of doing this, so long as there are no backdrop light sources.

Cloak of Darkness is a short and sweet game based on a light puzzle.

See **Room Descriptions** *for an item that can only be seen in bright light, when an extra lamp is switched on*

See **Looking Under and Hiding** *for a looking under action which is helped by the fiercer brightness of a light source*

See **Going, Pushing Things in Directions** *for making it hazardous to walk around in the dark*

See **Electricity and Magnetism** *for batteries to power a torch or flashlight*

See **Fire** *for a non-electrical way to produce light*

Start of Chapter 3: Place

Back to §3.6. Windows

Onward to §3.8. Sounds

Example 194: ★ **The Dark Ages Revisited** An electric light kind of device which becomes lit when switched on and dark when switched off.

Example 312: ★ **Hymenaeus** Understanding "flaming torch" and "extinguished torch" to refer to torches when lit and unlit.

Example 352: ★ **Reflections** Emphasizing the reflective quality of shiny objects whenever they are described in the presence of the torch.

Example 363: ★ **Peeled** Two different approaches to adjusting what the player can interact with, compared.

Example 26: ★★ **Down Below** A light switch which makes the room it is in dark or light.

Example 51: ★★ **The Undertomb 2** Flickering lantern-light effects added to the Undertomb.

Example 348: ★★ **Hohmann Transfer** Changing the way dark rooms are described to avoid the standard Inform phrasing.

Example 81: ★★★ **Unblinking** Finding a best route through light-filled rooms only, leaving aside any that might be dark.

Example 347: ★★★ **Zorn of Zorna** Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

Example 349: ★★★ **Four Stars 1** An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead.

Example 291: ★★★★ **Cloak of Darkness** Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

---

§3.8. Sounds

It is too easily assumed that room descriptions are what the player sees, but as **The Undertomb** demonstrates, they might just as easily include ambient sounds.

So Inform's "listening to" action is the audio equivalent of "examining", rather than "looking". Despite this the player can type LISTEN, which Inform understands as listening to the everything in the location at once. A simple but effective way to handle this is shown in **The Art of Noise**.

Four Stars 2 adjusts the idea of "visibility" to make it behave differently for listening purposes: this introduces a formal idea of "audibility".

See **Lighting** *for heightened hearing in darkness, and the rest of "Four Stars"*

Start of Chapter 3: Place

Back to §3.7. Lighting

Onward to §3.9. Passers-By, Weather and Astronomical Events

Example 50: ★ **The Undertomb 1** A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

Example 364: ★ **Four Stars 2** Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

Example 95: ★★★ **The Art of Noise** Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

---

§3.9. Passers-By, Weather and Astronomical Events

Out of doors, nature is seldom still. Clouds scull by at random, as in **Weathering**, and provide some variety in what would otherwise be lifelessly static room descriptions. In much the same way, passers-by and other diversions make a city street a constant bustle: see **Uptown Girls** for this human breeze. A more nagging sense of atmosphere can be experienced in **Full Moon**.

Orange Cones offers traffic that is present on every road in the story unless a room is marked off with orange cones -- and this is allowed to change during play.

Night and Day and **Totality** each schedule celestial events to provide a changing display in the sky above, and this time running like clockwork rather than at random.

★ *See* **Scene Changes** *for meteors and a moon-rise*

Start of Chapter 3: Place

Back to §3.8. Sounds

Onward to Chapter 4: Time and Plot: §4.1. The Passage Of Time

Example 131: ★ **Weathering** The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

Example 157: ★ **Full Moon** Random atmospheric events which last the duration of a scene.

Example 164: ★ **Night and Day** Cycling through a sequence of scenes to represent day and night following one another during a game.

Example 144: ★★ **Totality** To schedule an eclipse of the sun, which involves a number of related events.

Example 120: ★★★ **Orange Cones** Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

Example 132: ★★★ **Uptown Girls** A stream of random pedestrians who go by the player.

Examples from Chapter 3: Place

Start of this chapter

Chapter 4: Time and Plot

Indexes of the examples

A room which changes its description depending on whether an object has been examined.

Sometimes a nice effect is to change the way things are described depending on the information the player has gained in the course of play. We could for instance write this:

```
"Night Sky"

The Planetarium is a room. "[if we have examined the
sinister message]A dark room where it seems
something is about to jump out at you![otherwise]A
tranquil dark room with a ceilingful of stars.[end if]"

The sinister message is a thing in the Planetarium. "A
message is taped to the wall." The description is
"'BEWARE.'"

Test me with "look / x message / look".
```

On the other hand, beware that this would not work as desired:

```
"Night Sky"

The Planetarium is a room. "[if we have listened to the
sinister message]A dark room where it seems
something is about to jump out at you![otherwise]A
tranquil dark room with a ceilingful of stars.[end if]"

The sinister message is a thing in the Planetarium. "A
message plays very softly, so that you would have to
listen to hear it." Instead of doing anything other than
```

```
listening to the message: say "It's only a sound, after
all.". Instead of listening to the sinister message: say "A
voice whispers, 'BEWARE'."


Test me with "listen to message / look".
```

The reason is that our Instead rule has pre-empted normal listening, so Inform considers that we have never successfully heard the message. The moral here is that "if we have..." is useful for tracking events that otherwise proceeded completely normally (picking up ordinary objects, examining things); if we have used instead to make some change, we will have to use a different approach to record that the event did occur as scheduled.

ExampleInfiltration *WI*

A room whose description changes depending on the number of times the player has visited.

Suppose we have a location that makes the player uncomfortable, and we want its description to change slightly each time he goes there, to reflect his increasing unease. We also want the door to that room to show whether he is going there for the first time, or whether this is a repeat visit.

We start with an ordinary room:

```
"Infiltration"


The Wasteland is a room. "In its more distant reaches,
the Wasteland has a kind of austere beauty, but here
beside the Secure Zone it is the worst of all possible
worlds. Barrels of toxins are stacked the regulation
hundred and fifty feet out; more traditional garbage has
simply been flung over the wall, and this category
includes one or two corpses roughly and inadequately
disguised by black plastic bags. The wall itself has
become a canvas for outcasts and exiles, and is
covered with obscene paintings, lewd remarks about the
```

```
inhabitants of the Secure Zone, and a few maudlin
epitaphs."
```

Now the door, which will change from saying "leads inside..." to "leads back inside..." when this becomes appropriate:

```
The portal is a door. It is inside from the Wasteland and
outside from the Secure Zone. "[if the player is in the
Wasteland]To the west, a[otherwise]A[end if] portal in
the cinder-block and barbed wire wall leads[if the player
is in the Wasteland and the Zone is visited] back[end if]
[if the player is in the
Wasteland]inside[otherwise]outside[end if]."
```

Here we haven't used any conditions that we didn't know about in previous sections: the portal line only reflects whether the Zone has been visited never or visited once. But the Secure Zone itself makes use of the number of times visited:

```
The Secure Zone has the description "[if the player is in
the Zone for the second time]Re-entering the Zone has
not made you any more comfortable inside. [end
if]Despite your carefully-chosen outfit and the walk you
have been practicing, you are sure those inside can tell
you don't belong. Not that very many people are visible
here[if the player is in the Zone for more than the
second time] -- the place seems more desolate than
ever[end if]."
```

```
Instead of going west in the Wasteland, try going inside.
Instead of going east in the Secure Zone, try going
outside.
```

And finally, to be sure that the player does see our fancy changing descriptions:

```
Use full-length room descriptions.
```

```
Test me with "look / open portal / w / look / e / look / w /
```

e / w".

Notice that the description of the Secure Zone changes from visit to visit, but that looking repeatedly during a single visit changes nothing.

Example **Ant-Sensitive Sunglasses** *WI*

What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

Suppose we want to create an object -- or maybe even a series of objects -- that warp the player's perception of every room description and object around him.

We've already seen some ways to create variations in text. For instance, we could make a room description with if substitutions in it, like so:

```
The Kitchen is a room. "[if the player is wearing the
sunglasses]Are ants coming out of the sink? No,
probably no.[otherwise]A small kitchen tucked into a
corner of the vacation house. There is storage space for
five or six cups, a sink, a two-ring stove; nothing else to
speak of.[end if]"
```

That works fine if we have one or two variations we want to add; it's not so good if we're going to have several items that work like the sunglasses, or if we want the sunglasses to override the description of every room in the house.

A slightly more flexible method is to use a substitution that calls out to a say phrase, like this:

```
The Kitchen is a room. "[kitchen-description]"



To say kitchen-description:
    if the player is wearing the sunglasses:
        say "Are ants coming out of the sink? No, probably
no.";
    otherwise:
```

```
        say "A small kitchen tucked into a corner of the
    vacation house. There is storage space for five or six
    cups, a sink, a two-ring stove; nothing else to speak of."
```

But again this doesn't handle the case of overriding multiple rooms at once very well.

When we reach a point where we need a given piece of text to be very flexible depending on the world model, it's time to use an activity.

Activities offer several advantages. One, we can create an activity like this:

```
    Printing the room-description of something is an activity.
```

and then write a rule that applies to multiple rooms at once, like:

```
    Rule for printing the room-description of a room when
    the player wears the sunglasses:
        say "The walls look like they're covered with ants.
    Just a coincidence, I'm sure."
```

Inform's usual rule-ranking also means that more-specific rules will override less-specific ones, so we could add

```
        Rule for printing the room-description of the Kitchen
    when the player wears the sunglasses:
        say "Are ants coming out of the sink? No, probably
    not."
```

and have that rule override the behavior of the activity just in the kitchen. Meanwhile, our base room descriptions remain straightforward and uncluttered by if-statements.

Several other examples will show how to hook activities into existing actions: Crusoe goes into detail about how how to make the descriptions of things more variable, and Aftershock demonstrates activities for describing the behavior of switchable devices.

Here, we preview all of those methods, just to get a sense of how they work and why they might be useful in controlling a game. Subsequent chapters go into more detail about the syntax of creating activities and the list of activities that are already defined by Inform.

"Ant-Sensitive Sunglasses"

Part 1 - Procedure

To add a new activity to an existing Inform rule, we need to do three things:

1) Define our new activity.

2) Give a basic rule that says what is supposed to happen when that activity occurs, as in "Rule for..."

3) Replace the existing rule in Inform's rulebooks with a new one that calls on our activity.

Here we do this with examining:

Section 1 - Item Description

Printing the description of something is an activity.

Now, by default, we want to print the description property; we just want the option to write some extra rules overriding that property. So we tell Inform that our most basic rule for printing the description of something is just to give that description text:

```
Rule for printing the description of something (called
item):
    if the description of the item is not "":
        say "[description of item][paragraph break]";
    otherwise:
        say "You see nothing special about [the item].".
```

Next, we need the standard examining rule to look at our printing-the-description activity:

The activity-based examining rule is listed instead of the

```
standard examining rule in the carry out examining
rules.
```

```
This is the activity-based examining rule:
    carry out the printing the description activity with the
noun;
    rule succeeds.
```

Now we do the same thing to room descriptions.

```
Section 2 - Room Description
```

```
Printing the room-description of something is an activity.
```

```
Rule for printing the room-description of a room (called
item):
    if the description of the item is not "":
        say "[description of item][paragraph break]";
    otherwise:
        do nothing instead.
```

```
The activity-based room description body text rule is
listed instead of the room description body text rule in
the carry out looking rules.
```

Our replacement rule this time around is a little bit trickier just because the rule that we're replacing is a complicated one: describing a room already checks to see whether there's light to see by, whether the player has turned off room descriptions when he enters a room for the second time, and whether the player character is (say) inside a closed box he can't see out of.

But all of those details are re-copied from the standard rules, and the important thing is that, at the end, we again carry out our activity.

```
This is the activity-based room description body text
```

```
rule:
    if the visibility level count is 0:
        if set to abbreviated room descriptions, continue
the action;
        if set to sometimes abbreviated room descriptions
and
            abbreviated form allowed is true and
            darkness witnessed is true,
            continue the action;
        begin the printing the description of a dark room
activity;
        if handling the printing the description of a dark
room activity,
            say "It is pitch dark, and you can't see a thing.";
        end the printing the description of a dark room
activity;
    otherwise if the visibility ceiling is the location:
        if set to abbreviated room descriptions, continue
the action;
        if set to sometimes abbreviated room descriptions
and abbreviated form
            allowed is true and the location is visited,
continue the action;
        carry out the printing the room-description activity
with the location.
```

Section 3 - Device Description

```
Showing action of something is an activity.


Rule for showing action of something (called item):
    if the item is switched on, say "[The item] is switched
on.";
    otherwise say "[The item] is switched off."
```

The activity-based described devices rule is listed instead of the examine devices rule in the carry out examining rules.

This is the activity-based described devices rule:
    if the noun is a device:
        carry out the showing action activity with the noun;
        now examine text printed is true.

Report switching on something:
    say "You flip a switch. ";
    carry out the showing action activity with the noun instead.

Part 2 - Scenario

The Kitchen is a room. "A small kitchen tucked into a corner of the vacation house. There is storage space for five or six cups, a sink, a two-ring stove; nothing else to speak of."

The microwave is a device in the Kitchen.

South of the Kitchen is the Living Area. The description of the Living area is "A whitewashed living/dining/reclining area in what used to be a shepherd's stone hut, but now costs vacationers 600 euros a week. It offers no mod cons, only a straight view of the Mediterranean and a wobbly writing table."

Rule for printing the room-description of a room when the player wears the sunglasses:
    say "The walls look like they're covered with ants. Just a coincidence, I'm sure[antsy]."

Rule for printing the room-description of the Kitchen when the player wears the sunglasses:
    say "Are ants coming out of the sink? No, probably

not[antsy]."

Rule for printing the description of something (called the item) when the player wears the sunglasses:
    say "[The item] [are] [one of]ant-colored[or]ant-legged[or]covered in ants[at random][antsy]."

Rule for showing action of the microwave:
    say "The microwave hums meaningfully to itself."

Rule for showing action of the microwave when the player wears the sunglasses:
    say "The microwave hums as though inhabited by a billion ants[antsy]."

The player carries sunglasses of freakiness and an apple. The apple is edible. The sunglasses are wearable.

ant-paranoia is a number that varies.

To say antsy:
    increase ant-paranoia by 1;

Every turn:
    if the ant-paranoia is greater than 3:
        say "Augh! AUUUGH! GET THEM OFF--";
        end the story saying "You have lost your mind."

Test me with "look / turn on microwave / turn off microwave / x apple / x sunglasses / s / wear sunglasses / look / x apple / n / turn on microwave".

Example **Rip Van Winkle**

A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

There are times when, for greater elegance of prose, we'd like to mention an object in the main body text of a room. For instance:

```
Here is a lovely, secluded fold in the mountains, far from
civilization: as though to prove it, Rip Van Winkle is
sleeping under a tree.
```

As we've already seen, that's no problem if Rip is scenery. He'll stay there motionless.

But what if something in the game allows Rip to wake up? Or what if we want to use the same technique on a portable object that the player should be allowed to take? Clearly in that case it's not appropriate to make the mentioned thing be scenery, and at the same time, we need to keep Inform from adding a superfluous

```
You can see Rip Van Winkle here.
```

to the end of our description.

Here is how:

```
"Rip Van Winkle"
```

```
A person can be asleep.
```

```
The Catskills is a room. "Here is a lovely, secluded fold
in the mountains, far from civilization[if Rip Van Winkle
is asleep]: as though to prove it, Rip Van Winkle is
sleeping under a tree[end if]."
```

A tree is scenery in the Catskills.

Rip Van Winkle is a man in the Catskills. Rip Van Winkle is asleep.

Before listing nondescript items of the Catskills:
    if Rip Van Winkle is marked for listing:
        now Rip Van Winkle is not marked for listing;
        if Rip Van Winkle is not asleep,
            say "Rip Van Winkle stands here, looking mightily confused."

Instead of waiting:
    say "Rip Van Winkle wakes up with a snort.";
    now Rip Van Winkle is not asleep.

Test me with "look / z / look".

---

358 ExamplePriority Lab
A debugging rule useful for checking the priorities of objects about to be listed.

When it comes time to start manipulating the priorities of items, it is useful to be able to check the table for debugging purposes; the problem is that printing the names of the objects can itself affect the way the room description is generated, foiling our debugging efforts.

What follows is a rule to help with debugging safely, and a sample of how priorities work:

"Priority Lab"

```
Section 1 - Procedure


Before printing the locale description (this is the dump
locale table rule):
    say "Locale Priority list:";
    repeat through Table of Locale Priorities:
        let the flag be whether or not the notable-object
entry is mentioned;
        say "[line break] [notable-object entry]: [locale
description priority entry]";
        if the flag is false, now the notable-object entry is
not mentioned;
    say line break.
```

Now, let's look at some items put in a specific order. Things with low priority numbers list towards the beginning; things with high priority numbers list towards the end. (It helps to think of it as though we were making a numbered list of the paragraphs to appear in the description.) Anything numbered 0 doesn't appear at all, and the default priority of an object is 1.

```
A thing can be early-described, late-described, latest-
described, never-described, sightline-described, or
ordinarily-described. A thing is usually ordinarily-
described.


After choosing notable locale objects (this is the apply
early and late description rule):
    repeat with item running through early-described
things:
        if there is a notable-object of item in the Table of
Locale Priorities:
            set the locale priority of the item to 1; [list before
everything else -- this would work with any number
lower than 5 and higher than 0]
    repeat with item running through late-described
things:
        if there is a notable-object of item in the Table of
Locale Priorities:
            set the locale priority of the item to 10; [list after
everything else -- this would work with any number
```

```
larger than 5]
    repeat with item running through never-described
things:
        set the locale priority of the item to 0; [don't list at
all]
    continue the activity.
```

An important cautionary note: priorities are only honored if the objects are going to get their own paragraphs (with "writing a paragraph about..." or because they have initial appearances). Priorities do not affect the order in which items appear in the final "You can see..." list, except that items with priority 0 or lower are omitted. (If we want to order the items in that list, we may want to resort to the Complex Listing extension by Emily Short.)

There are further refinements available to us: for instance, we could make some things that are only visible if the player is raised above ground level.

```
After choosing notable locale objects (this is the
sightline-described things are visible from supporters
rule):
    if the player is not on a supporter:
        repeat with item running through sightline-
described things:
            if there is a notable-object of item in the Table of
Locale Priorities:
                set the locale priority of the item to 0; [remove
objects that can only be seen from higher objects.]
    continue the activity.
```

It may also be useful to know about the "parameter-object", which refers to the thing whose contents we are currently describing: the standard rules consider how to describe the contents of the location and then also check the contents of any supporter or container the player may be inside, so in the first case "parameter-object" would be the location, and then in the second the supporter in question.

In practice this is rarely useful, but should we need to change priorities in the case of both player and object being inside a particular container, we might make use of it, for instance:

```
A thing can be tasteful or icky. A thing is usually tasteful.
```

```
After choosing notable locale objects (this is the icky
things next to players rule):
    if the player is on the parameter-object:
        repeat with item running through icky things :
            if there is a notable-object of item in the Table of
Locale Priorities:
                set the locale priority of the item to 10;
[remove objects that can only be seen from higher
objects.]
    continue the activity.
```

The other thing to note is that by default that final collection of generic objects ("You can also see...") appears at the end, regardless of the priority of everything else. If we really wanted to, though, we could force something to appear even after that paragraph, by adding a new listing rule to the locale description rules:

```
After choosing notable locale objects (this is the latest-
described items priority rule):
    repeat with item running through latest-described
things:
        if the item is a notable-object listed in the Table of
Locale Priorities:
            now the item is mentioned;
            now the item is marked for late listing.


The late listing rule is listed after the you-can-also-see
rule in the for printing the locale description rules.


A thing can be marked for late listing. A thing is usually
not marked for late listing.


This is the late listing rule:
    if something is marked for late listing:
        say "Oh! And also [a list of things which are
marked for late listing].";
        now everything is not marked for late listing;
    continue the activity.
```

```
Section 2 - Scenario


The Priority Lab is a room. The early bird, the worm, the
leaf, the unseen object, the pebble, the twig, and the
late edition are things in the Priority Lab.


The early bird is early-described. The late edition is late-
described. The unseen object is never-described.


The worm is icky.


The high window is in Priority Lab. It is sightline-
described and fixed in place. The initial appearance of
the high window is "There's a tiny high window up near
the ceiling that you can't see unless you're on top of
something."
```

In order for the priorities we just set to be interesting, let's give out some initial appearances and writing a paragraph rules:

```
The initial appearance of the worm is "A worm inches
along the ground."
The initial appearance of the late edition is "Finally, the
late edition lies at your feet."


Rule for writing a paragraph about the early bird when
the early bird is in a room: say "The early bird always
appears first, and here it is."
Rule for writing a paragraph about the leaf: say "Look,
there's [a leaf][unless the leaf is in the location] on [the
holder of the leaf][end if]!"
Rule for writing a paragraph about an icky thing (called
icky item) which is on something which supports the
player: say "Ew, [an icky item] is right next to you."
```

This procedure also means (as you can test by experiment) that after the

late edition has been picked up and dropped again, it lists in no special order in the "you can see..." paragraph (since initial appearances only print when the object has not yet been moved).

```
The afterthought is a thing in the Priority Lab. It is latest-
described.


The bar stool is an enterable supporter in Priority Lab.


Test me with "get leaf / drop leaf / look / x unseen object
/ get pebble / look / get twig / look / get afterthought /
look / drop twig / look / get late edition / look / drop late
edition / sit on bar stool / look / get all / put all on stool /
look".
```

Example **Low Light**

An object that is only visible and manipulable when a bright light fixture is on.

Suppose we want a different treatment of lighting than the usual: the room isn't totally dark, but there's something we can't see unless we turn on a bright light.

```
"Low Light"
```

First we make our environment and its light:

```
The Workroom is a room. The desk is in the Workroom.
The brilliant lamp is a device on the desk.


To decide whether the light level is high:
    if the brilliant lamp is switched off, no;
```

```
    if the player cannot see the brilliant lamp, no;
    yes.


To decide whether the light level is low:
    if the light level is high, no;
    yes.
```

Now we make a shadow so that the player can only refer to it if the shadow is in inventory or the light is on:

```
The shadow is a privately-named thing on the desk.


Understand "barely-visible" or "barely visible" or
"shadow" as the shadow when the light level is high.
Understand "invisible" or "shadow" as the shadow when
the player encloses the shadow.
```

And finally a couple of extra touches to make it clear why we're able to interact with the shadow when it's in inventory, even if the light is low:

```
Before printing the name of the shadow:
    if the light level is high:
        say "barely-visible ";
    otherwise if the player encloses the shadow:
        say "invisible (but tangible) "


After dropping the shadow when the light level is low:
    say "You let it go and it fades into the ambient
gloom."
```

To handle the appearance of the object, we want to set its locale priority to 0: that will prevent it being named in room descriptions.

```
After choosing notable locale objects:
    unless the light level is high:
        set locale priority of the shadow to 0.
```

Test me with "look / get shadow / turn on lamp / look / get shadow / i / turn off lamp / i / drop shadow / look / get shadow / turn on lamp / look".

Example **Slightly Wrong** *WI*

A room whose description changes slightly after our first visit there.

A fairly common effect in interactive fiction is a room which is described differently on the first visit than on subsequent visits. We can produce this effect as follows:

"Slightly Wrong"

Awning is a room. "A tan awning is stretched on tent poles over the dig-site, providing a little shade to the workers here; you are at the bottom of a square twenty feet on a side, marked out with pegs and lines of string. Uncovered in the south face of this square is an awkward opening into the earth."

Slightly Wrong Chamber is south of the Awning. "[if unvisited]When you first step into the room, you are bothered by the sense that something is not quite right: perhaps the lighting, perhaps the angle of the walls. [end if]A mural on the far wall depicts a woman with a staff, tipped with a pine-cone. She appears to be watching you."

Test me with "look / s / look".

Note the "[if unvisited]..." in the description of the Slightly Wrong Chamber. A room is considered to be "unvisited" until after the player has seen its description for the first time.

The bracketed text creates a special rule for printing; we will learn more about these in the sections on text with variations and text with substitutions.

Some further fine print: we might write our condition as "if unvisited", "if the location is unvisited", or "if the Chamber is unvisited" -- all of these constructions would be acceptable, but in the absence of more specifics, the condition is understood to apply to the object whose description it is.

---

357 **Example** **The Eye of the Idol** *WI*

A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

"The Eye of the Idol"

Section 1 - Reusable Material

We start by defining relations that let us know where items "belong", with the understanding that if something is where it belongs, it will be described in the main room description and therefore should not be separately listed. Thus:

Positioning relates various things to various things. The verb to be placed in means the positioning relation. The verb to be placed on implies the positioning relation.

Room-positioning relates various things to various rooms. The verb to be room-placed in means the room-positioning relation.

We can't make relations relate various objects to various objects, and rooms are not things, so two separate cases are necessary. An alternative approach would be to say "A thing has an object called the initial placement", which would allow a thing to have an initial placement that was a room, a supporter, or a container; an advantage of using

relations, though, is that that way we can if we like specify multiple placements for the same object, so that, e.g., a sparkling diamond can be described in the main description paragraph as "half-buried in dust" in the beginning of the game, and then at the end as "in the eye of the idol" at the end.

Now we define, based on these relations, an "in-place" adjective, which will identify whether something is in a location which will specially describe it:

```
Definition: a thing (called prop) is in-place:
    if the prop is in the location and the prop is room-
placed in the location, yes;
    if the holder of the prop is a thing and the prop is
placed in the holder of the prop, yes;
    no.
```

```
Definition: a thing is out-of-place if it is not in-place.
```

With that done, removing these items automatically from the room description is actually pretty easy:

```
Before listing nondescript items:
    now every marked for listing in-place thing is not
marked for listing.
```

One tricky case remains: when something is placed on a supporter that is scenery, it can be mentioned even if we have marked that object "not marked for listing". What matters here is not whether the object itself is marked for listing but whether the supporter has been "mentioned". (A fuller description of how room descriptions are assembled is available in the Looking section of the Commands chapter in the Recipe Book.) So let's also add a feature whereby we can easily suppress the descriptions of these supporters when appropriate:

```
A supporter can be quiet.
```

A quiet supporter is one that is never mentioned itself and which only mentions its contents if they are out of place. This allows for maximum flexibility in incorporating it into the body of room descriptions.

```
Rule for writing a paragraph about a quiet supporter
(called chosen table):
    if an out-of-place thing is on the chosen table:
        if an in-place thing is on the chosen table,
            say "On [the chosen table], in addition to [the list
of in-place things on the chosen table], [is-are a list of
out-of-place things which are on the chosen table].";
        otherwise say "On [a chosen table] [is-are a list of
out-of-place things which are on the chosen table].";
    now the chosen table is mentioned.
```

Notice that we can still override this with writing a paragraph rules about specific supporters in our game, if we decide that we want something a little different in some cases.

Now, an example to test this out:

```
Section 2 - A Sample Scenario

The Sand-Floored Chamber is a room. "The constant
wind has filled this chamber with a layer of fine red
sand, as soft as powder snow[if the diamond is in the
Sand-floored Chamber]. Something sparkling is half-
buried in the corner[end if]. A doorway lies open to the
north."

The sparkling diamond is in the Sand-floored Chamber.
The sparkling diamond is room-placed in the Sand-
floored Chamber. The description is "It is a vast
diamond; the front is faceted, the back smoothed to fit in
some sort of socket."

The Hexagonal Temple is north of the Sand-Floored
Chamber. "The temple walls are great ashlar blocks
rising to a hundred feet overhead, perhaps more; the
roof is a scarlet awning only, through which the sun
filters down in blood hues. Overseeing all is a sculpture
in stone and ivory[if the sparkling diamond is in the idol's
eye], in whose single eye a vast diamond gleams[end if]
```

```
[mat-and-incense text].".
```

```
To say mat-and-incense text:
    if the mat is in the Temple and the incense stick is on
the pedestal:
        say ". A prayer mat at the idol's feet, and an
incense stick still burning on the pedestal, indicate that
someone was only recently consigning her grievances
to the care of the deity";
    otherwise if the mat is in the Temple:
        say ". At the idol's feet, some worshipper has left a
prayer mat";
    otherwise if the incense stick is on the pedestal:
        say ". At the idol's side is a pedestal, on which
incense still smolders".
```

We could have done all this with text conditions in the main room description, but it becomes difficult to read when there are too many conditions operating in the same text property, so we break it out into a clearer set of conditions.

```
The idol is scenery in the Hexagonal Temple.
Understand "sculpture" or "stone" or "ivory" as the idol.
The description is "The idol is perhaps three times the
height of an ordinary man."
```

```
The idol's eye is part of the idol. It is a container. The
description is "[if the diamond is in the idol's eye]It
gleams with purpose and righteous wrath[otherwise]A
round socket in the center of the idol's forehead from
which something seems to be missing[end if]."
```

```
The pedestal is a quiet supporter in the Hexagonal
Temple. On the pedestal is an incense stick. The
incense stick is placed on the pedestal.
```

```
A mat is in the Hexagonal Temple. It is room-placed in
the Hexagonal Temple. The description is "Woven of
assorted grasses."
```

Test me with "get diamond / look / n / get mat / look / drop diamond / look / get diamond / put diamond in eye / look / get incense / look / drop mat / look / get mat / put mat on pedestal / look / put incense on pedestal / look".

---

362 **ExampleCopper River** *WI*

Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

In a very dense environment, we might want to offer the player room descriptions in which only the currently-interesting items are mentioned, while other objects are suppressed even if they are present. In effect, this takes the idea of scenery and makes it more flexible: different things might become background objects or foreground objects at different times during play.

There are a wide range of possible reasons to do this -- to shift the narrative emphasis, to change the mood of the game by highlighting different parts of the environment, to show the game from the perspective of different viewpoint characters -- but in the following example, our goal is to show the player only the objects that are currently useful for puzzles.

To do this, we need some notion of what puzzles are currently available and unsolved, so we make an "unsolved" adjective; we also need to know which things solve the puzzle, so we create a "resolving" relation, to indicate which objects resolve which problems.

Given that information, we can create rules about which objects in the game world are currently interesting, which are currently dull, and describe accordingly:

"Copper River"

Use scoring.

Section 1 - Procedure

Resolving relates various things to various things. The verb to resolve means the resolving relation.

Definition: a thing is interesting if it is not dull.

Definition: a person is dull:
    no.

Definition: a thing is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    yes.

Definition: a supporter is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    if it supports an interesting thing, no;
    yes.

Definition: a container is dull:
    if it is unsolved, no;
    if it resolves an unsolved thing, no;
    if it contains an interesting thing, no;
    yes.

After choosing notable locale objects:
    repeat with item running through unsolved things:
        set the locale priority of the item to 1.

For printing a locale paragraph about a dull thing (called item):
    now the item is mentioned.

```
Before printing a locale paragraph about a supporter
(called item):
    now every dull thing on the item is mentioned.


Before printing a locale paragraph about a container
(called item):
    now every dull thing on the item is mentioned.


Instead of searching a supporter:
    if the noun supports something interesting:
        say "[A list of interesting things on the noun] [are]
on [the noun]";
        if the noun supports something dull:
            say " (alongside [a list of dull things on the
noun])";
        say ".";
    otherwise if the noun supports something dull:
        say "There's nothing very useful here, only [a list of
dull things on the noun].";
    otherwise:
        say "[The noun] [are] completely bare."


Instead of searching a container:
    if the noun contains something interesting:
        say "[A list of interesting things in the noun] [are] in
[the noun]";
        if the noun contains something dull:
            say " (alongside [a list of dull things in the
noun])";
        say ".";
    otherwise if the noun contains something dull:
        say "There's nothing very useful here, only [a list of
dull things in the noun].";
    otherwise:
        say "[The noun] [are] completely empty."


Before listing contents when not taking inventory: group
dull things together.
```

Rule for grouping together dull things: say "assorted dull items".

Section 2 - Scenario World and Objects

The Kitchen is a room. "Your Aunt Fiona's kitchen looks as though it has been at the eye of a glitter storm. Fine, sparkling grit dusts every surface. The appliances are slightly askew, too, as though they hadn't quite settled after a vigorous earthquake."

The shelf is a scenery supporter in the Kitchen. On the shelf is a can of beans, a can of potato leek soup, and a tin of deflating powder.

The cabinet is a scenery container in the Kitchen. In the cabinet is a book of matches, a bottle of descaling solution, a fish hook, and a rusty knife. It is openable and closed.

The counter is a scenery supporter in the Kitchen. On the counter is an espresso machine, a blender, and a mortar. The blender and the mortar are containers. In the mortar is a pestle. Understand "countertop" as the counter.

The stove is a scenery supporter in the Kitchen. The oven is part of the stove. The oven is a closed openable container.

The refrigerator is a fixed in place container in the Kitchen.

Understand "fridge" as the refrigerator.

The description is "The refrigerator is a dull blue-green, and has a puffy, marshmallow texture on the outside, which means that it's no good for sticking magnets to. Aunt Fiona has never been willing to explain where she got it." The refrigerator is openable and closed.

In the refrigerator are a bottle of ice wine, a bag of carrot sticks, and an egg.

Aunt Fiona is a woman in the Kitchen. Aunt Fiona can be inflated or deflated. Aunt Fiona is inflated. "[if Aunt Fiona is inflated]Aunt Fiona stands nearby. Or perhaps 'stands' is the wrong word: she has been sort of puffed up in her own skin like a balloon, and is now propped in a corner of the room with her head lolling back[otherwise]Aunt Fiona stands -- on her own two slender legs -- at the center of the room[end if]."

Every turn when Fiona is unsolved and Fiona can see the player:

if a random chance of 1 in 3 succeeds:

say "[one of]Aunt Fiona's eyes follow you, wide and desperate, but it doesn't look like she's able to do anything[or]Aunt Fiona is still looking reproachful[or]A faint gurgling comes from Aunt Fiona[or]Aunt Fiona makes a funny croak noise[or]Aunt Fiona is still having trouble speaking. Perhaps her throat is as swollen as the rest of her[or]Aunt Fiona twitches[stopping]."

There is a thing called a salmon. Understand "fish" as the salmon. The salmon can be scaly or prepared. The salmon is scaly. The description is "[if scaly]It looks delicious, but is still covered with scales[otherwise]The salmon has been scaled and is ready to eat[end if]."

Before printing the name of the salmon when the salmon is scaly:

```
    say "very scaly ".
```

Section 3 - Scenario Puzzles

```
Definition: Aunt Fiona is unsolved if she is inflated.
Definition: the salmon is unsolved:
    if the salmon is off-stage, no;
    if the salmon is scaly, yes;
    no.
```

The deflating powder resolves Aunt Fiona.

```
Instead of putting the deflating powder on Aunt Fiona:
    try throwing the deflating powder at Aunt Fiona.
Instead of giving the deflating powder to Aunt Fiona:
    try throwing the deflating powder at Aunt Fiona.
```

```
Instead of throwing the deflating powder at Aunt Fiona:
    if Aunt Fiona is inflated:
        say "You toss some of the powder in Aunt Fiona's
direction, and with a sudden gaseous HUFF! she
returns to her usual shape and size. [paragraph
break]'Well!' she says, brushing herself off. 'That was
bracing!' [paragraph break]You give her an
embarrassed smile, to apologize for not curing her
faster.";
        now Aunt Fiona is deflated;
        increase the score by 2;
    otherwise:
        say "[one of]You throw another hefty dose of the
powder at your aunt. [paragraph break]'Thank you,
child,' she says, sneezing. 'But I think you've done
enough now.'[or]You throw another hefty dose of the
powder at your aunt. [paragraph break]'You're too kind,'
she wheezes, through a cloud of glittering dust.
[or]You've probably done enough with the powder.
[stopping]".
```

Every turn when Aunt Fiona is deflated and the salmon is off-stage:
    move the salmon to the counter;
    say "'At least they didn't get this,' she says, producing from somewhere on her person a fresh-caught salmon. An odd pattern around its eye sockets makes it looks comically as though it wears spectacles. 'It's the Salmon of Knowledge,' she explains casually. 'We just need to scale and cook it.'"

The bottle of descaling solution resolves the salmon.

Does the player mean putting the descaling solution on the fish hook: it is unlikely.
Does the player mean putting the descaling solution on the salmon: it is very likely.

Instead of putting the bottle of descaling solution on the salmon:
    if the salmon is scaly:
        now the salmon is prepared;
        say "With just a single squirt of the descaling solution (which confusingly has a picture of bathroom tiles on the label), you remove the scales from the salmon, leaving its pink flesh ready for preparation.";
        increase the score by 2;
    otherwise:
        say "'Don't do that,' Aunt Fiona warns you. 'Excessive applications could damage the flesh.'"

Test me with "look / get powder / drop powder / look / look in cabinet / get powder / put powder on fiona / look / open cabinet / look in cabinet / get solution / open fridge / put solution in fridge / look / get solution / put solution on salmon / look".

Example **Port Royal 1**

A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

"1691"

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Unless we arrange otherwise, this will be the first room in the game because it is the first we have defined.

For subsequent rooms, we do not have to say explicitly that they are rooms, as long as they are connected to a room on the map. For instance, this will automatically make Thames Street End a room:

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east.

Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in

```
Port Royal from its earliest days, it is nearly the largest
building in the town."
```

If we have some concern that the room name will be confused with an existing name, we can be more explicit about it using "called":

```
East of Water Lane is a room called Thames Street at
the Wherry Bridge. Thames Street at the Wherry Bridge
has the description "To the southwest is the fishmarket;
directly across the street is the entrance to a private
alley through a brick archway."
```

```
The Private Alley is south of Thames Street at the
Wherry Bridge. "You're just outside the tavern the
Feathers. To the north, under a pretty little archway, is
the active mayhem of Thames Street, but the alley
narrows down to a dead end a little distance to the
south."
```

And now we get "inside", which generates a space treated as its own area on the map.

```
The Feathers is inside from the Private Alley. "Newly
built with brick, replacing the older Feathers tavern that
used to stand here. It sells wines in quantity, as well as
serving them directly, and the goods are always of the
best quality. There's a room upstairs for those wanting
to stay the night." The Feathers Bedroom is above the
Feathers.
```

And if we like we can declare a number of rooms for which we will come back and write the descriptions later. There is no obligation for the description to occur at the first definition of the room.

```
Lime Street is south of Thames Street End.
```

For efficiency, we can also write multiple sets of connections at once:

```
Queen Street East is east of Queen Street Middle and
```

south of Private Alley.

Clicking Go will translate this description into a sketchy but working simulation of Port Royal, in which we can type movement commands like EAST or SOUTH to explore the streets. Looking at the World tab of the Index, we can also see a schematic map of the simulation as it currently stands. Like the rest of the Index, this is provided entirely for the author's benefit, and is not visible to the player. (Though if we do decide that we want players to have access to a printed map while they play, Inform can help: we will return to the layout of Port Royal in the chapter on Publishing.)

The following Test command allows us to type TEST ME and explore the map we just devised:

Test me with "s / e / e / s / in".

Another part of Port Royal, with less typical map connections.

"1691"

Thames Street End is a room.

If we check out a map of historic Port Royal, we find that Thames Street End bends around the northwest tip of the peninsula and becomes the (very) roughly north/south Fisher's Row. We can't put Fisher's Row south of Thames Street End, though, because Lime Street is already going that way. So instead, let's have a map connection that bends around from west to north:

West of Thames Street End is north of Fisher's Row.

Now continuing west along Thames Street, or north along Fisher's Row, will bring us around the corner in question. Asymmetric map connections

should be used carefully. They're good for representing the layout of the real world, which tends not to be laid out on a convenient square matrix, but if exits are not described clearly they can be disorienting for the player. So let's be sure to make things clear:

```
The description of Fisher's Row is "A waterfront street
that runs south towards Chocolata Hole, where the
small craft are harboured. It also continues north around
the tip of the peninsula from here, turning into the east-
west Thames Street."
```

Meanwhile, suppose Fort James is in a prominent position, raised a bit from its surroundings; maybe the player should be able to go down from there, as well as south, to get to Thames Street End.

```
Thames Street End is down from Fort James. Thames
Street End is south from Fort James.
```

But we don't want the upward direction to work:

```
Up from Thames Street End is nowhere.
```

```
Test me with "n / d / u / w / e / n / s".
```

ExamplePort Royal 3

Division of Port Royal into regions.


```
"1691"
```

We should go ahead and do all our room definitions first...

Fort James is a room. "The enclosure of Fort James is a large, roughly hexagonal court walled with heavy stone. The walls face the entrance to Port Royal Harbour, and the battery of guns is prepared to destroy any enemy ship arriving."

Thames Street End is south of Fort James. "The ill-named Thames Street runs from here -- at the point of the peninsula -- all the way east among houses and shops, through the Fish Market, edging by the round front of Fort Carlisle, to the point where the town stops and there is only sandy spit beyond. Most of that stretch is full of people at all hours. Imported goods are moved off of ships and taken to distributors; exported goods are brought to be loaded; and there is one public house and brothel for every ten inhabitants.

Lime Street, wider and healthier but not as rich, runs directly south, and to the north the road opens up into the courtyard of Fort James."

Lime Street is south of Thames Street End. West of Thames Street End is north of Fisher's Row. The description of Fisher's Row is "A waterfront street that runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west Thames Street."

Thames Street End is down from Fort James. Up from Thames Street End is nowhere.

Water Lane is east of Thames Street End. "Here Thames Street -- never very straight -- goes steeply southeast for a portion before continuing more directly to the east.

Water Lane runs south toward Queen Street, and facing onto it is the New Prison -- which, in the way of these things, is neither. It did serve in that capacity for a time, and in a measure of the villainy which has been usual in Port Royal from its earliest days, it is nearly the largest building in the town."

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Fishmarket is southwest of Thames Street at the Wherry Bridge.

The Private Alley is south of Thames Street at the Wherry Bridge. "You're just outside the tavern the Feathers. To the north, under a pretty little archway, is the active mayhem of Thames Street, but the alley narrows down to a dead end a little distance to the south."

The Feathers is inside from the Private Alley. "Newly built with brick, replacing the older Feathers tavern that used to stand here. It sells wines in quantity, as well as serving them directly, and the goods are always of the best quality. There's a room upstairs for those wanting to stay the night." The Feathers Bedroom is above the Feathers.

Thames Street by the King's House is east of Thames Street at the Wherry Bridge. "The King's House is reserved for the use of the Governor, but he does not live in it, and it is frequently being rented out to some merchant so that the government will at least derive some value from it. It is nearly the least interesting establishment on Thames Street, and the crowd -- which, to the west, is extremely dense -- here thins out a

bit."

Thames Street before Fort Carlisle is east of Thames Street by the King's House. "Here Thames Street, formerly a respectable width, narrows to a footpath in order to edge around the front of Fort Carlisle, underneath the mouths of the cannon.

There are no buildings on the harbour side of Thames Street at this point, which means that you have an unusually good view of the ships at dock, water beyond, and the Blue Mountains rising on the other side of the harbour."

South of Thames Street before Fort Carlisle is a room called Fort Carlisle. The description of Fort Carlisle is "Handsomely arrayed with cannons which you could fire at any moment -- though of course there are ships at dock which might be in the way."

Queen Street End is south of Lime Street.

Queen Street Middle is east of Queen Street End.

Queen Street East is east of Queen Street Middle and south of Private Alley.

Queen Street at the Prison is east of Queen Street East.

Now, if we like, we can create regions to distinguish the coast from the portions of town that aren't on the water:

```
Inland is a region. Queen Street End, Queen Street
Middle, Queen Street East, Private Alley, Lime Street,
and Queen Street at the Prison are in Inland.
```

```
Waterfront is a region. Thames Street before Fort
Carlisle, Thames Street by the King's House, Thames
Street at the Wherry Bridge, Water Lane, Fishmarket,
Fisher's Row, and Thames Street End are in Waterfront.
```

There's no rule that regions must be contiguous, so we could if we like make a region consisting just of the two forts:

```
Military Holdings is a region. Fort Carlisle and Fort
James are in Military Holdings.
```

And we might make the Feathers Tavern part of the Inland area, but within its own subcategory:

```
Tavern is a region. It is in Inland. Feathers and Feathers
Bedroom are in Tavern.
```

Now the index map will be colored to reflect our regions, and later in the game development we would be able to make rules that affect just one region at a time.

Example **All Roads Lead to Mars** *WI*

Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

Suppose we want to allow the player to wander freely in any direction, but ourselves maintain control over the order in which he encounters the rooms. This sort of effect emphasizes the order of the story-telling over any kind of rigorous simulation of space; on multiple play-throughs, the player might not find all the same rooms in the same locations.

```
"All Roads Lead to Mars"
```

```
Before going a direction (called way) when a room
(called next location) is not visited:
    let further place be the room the way from the
location;
    if further place is a room, continue the action;
    change the way exit of the location to the next
location;
    let reverse be the opposite of the way;
    change the reverse exit of the next location to the
location.


The Open Plain is a room. "A wide-open grassy
expanse, from which you could really go any way at all."


The Hilly Place is a room. "The grassland gives way to a
somewhat more hilly area, though there is still very little
to guide you any particular way."


The Stream is a room. "This is the third place you've
been today, and so the stream is welcome. How
refreshing!"


Test me with "n / s / e / e".
```

If we wanted still to be able to find routes between places, we could define a relationship of connection between rooms, which we would add to as we went along.

ExampleBee Chambers

A maze with directions between rooms randomized at the start of play.

WI

Mazes are a traditional element of interactive fiction, often consisting of apparently identical rooms with exits that do not work reciprocally and

which cause confusion.

The methods of mapping mazes are now fairly well understood and mazes themselves tend to be regarded as tiresome rather than enjoyable by a large portion of the playing audience. However, if we did want to ignore the common wisdom and create a maze, randomly generated at the start of play, here would be one way to go about it:

```
"Maze of Gloom"


A Bee Chamber is a kind of room. The printed name of
a Bee Chamber is usually "Hexagonal Room". The
description of a Bee Chamber is usually "Waxy,
translucent walls surround you on six sides; the floor
and ceiling are made of the same material, gently
uneven. There are exits in every direction, cut into the
faces or the corners."


Bee1, Bee2, Bee3, Bee4, Bee5, Bee6, Bee7, Bee8,
Bee9, and Bee10 are Bee Chambers.


When play begins:
    now right hand status line is "[number of visited
rooms]/[number of rooms]";
    repeat with place running through Bee Chambers:
        now a random Bee Chamber is mapped north of
place;
        now a random Bee Chamber is mapped northwest
of place;
        now a random Bee Chamber is mapped west of
place;
        now a random Bee Chamber is mapped southwest
of place;
        now a random Bee Chamber is mapped south of
place;
        now a random Bee Chamber is mapped southeast
of place;
        now a random Bee Chamber is mapped east of
place;
        now a random Bee Chamber is mapped northeast
of place;
```

```
        now a random Bee Chamber is mapped above
    place;
        now a random Bee Chamber is mapped below
    place;
        now a random Bee Chamber is mapped inside
    place;
        now a random Bee Chamber is mapped outside
    place.



    Test me with "in / out / up / down / n / ne / nw / e / w / sw
    / se / s".
```

Example Zork II 134

A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby.

All we need to do is select the player's destination for him at random:

```
    "Zork II"



    The Carousel Room is a room.



    Instead of going from the Carousel Room:
        move the player to a random adjacent room.
```

To avoid infringing the original game too much, let's try a somewhat different setting:

```
    The Games of Chance is north of the Carousel Room.
    The Haunted Funhouse is northwest of the Carousel
    Room. The Ferris Wheel is east of the Carousel Room.
    The Topsy-Turvy is northeast of the Carousel Room.
    The Reproduction Henge is south of the Carousel
    Room. The Women's Toilet is southwest of the Carousel
```

```
Room. The Men's Toilet is southeast of the Carousel
Room. The Cotton Candy Shop is west of the Carousel
Room.
```

```
Test me with "s".
```

And the following means that the test runs consistently even though the numbers are theoretically random. To make them truly random, remove this line.

```
When play begins, seed the random-number generator
with 1234.
```

Or if we want to add the refinement that the Carousel Room can be switched off:

```
"Zork II"
```

```
The Carousel Room is a room. The spinning machine is
a switched on device in the Carousel Room.
```

And then

```
Instead of going from the Carousel Room when the
spinning machine is switched on:
    move the player to a random adjacent room.
```

```
The Games of Chance is north of the Carousel Room.
The Haunted Funhouse is northwest of the Carousel
Room. The Ferris Wheel is east of the Carousel Room.
The Topsy-Turvy is northeast of the Carousel Room.
The Reproduction Henge is south of the Carousel
Room. The Women's Toilet is southwest of the Carousel
Room. The Men's Toilet is southeast of the Carousel
Room. The Cotton Candy Shop is west of the Carousel
Room.
```

Test me with "turn off machine / s / n / turn on machine / s".

When play begins, seed the random-number generator with 1234.

ExampleIndirection

WI

Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

In Mayan culture, colours seem to have been used as names for the primary directions: for instance, "red" implies east as the colour of sunrise. So the following might be a stylish touch for a game in which the player has to get inside the Mayan world-view:

"Indirection"

Understand "white" and "sac" as north. Understand "red" and "chac" as east. Understand "yellow" and "kan" as south. Understand "black" and "chikin" as west.

We could also use a colour as a verb:

Understand "turquoise" and "yax" as looking.

And now a few extra rooms to try it out in:

The Square Chamber is a room. "A sunken, gloomy stone chamber, ten yards across. A shaft of sunlight cuts in from the steps above, giving the chamber a diffuse light, but in the shadows low lintelled doorways to east and south lead into the deeper darkness of the

Temple."

The Wormcast is east of the Square Chamber. The Corridor is south of the Square Chamber.

Test me with "kan / white / chac / black".

ExamplePrisoner's Dilemma

A button that causes a previously non-existent exit to come into being.

We can change the directions in the map in mid-game, though in practice this is rarely necessary. But suppose we do not want a door or any sign of a door to exist before the player takes some action, in this case pressing a button:

"Prisoner's Dilemma"

Challenger's Waiting Room is a room. "The challenge is this: to wait as long as you can endure to do so in a room with no features and no clock. If you wait longer than all the other contestants, you win."

The button is fixed in place in the Challenger's Waiting Room. "The only item in view is a black recessed button."

Amid the Cheering Throng is a room.

```
Instead of pushing the button for the first time:
    change the east exit of the Challenger's Waiting
Room to Amid the Cheering Throng;
    change the west exit of the Cheering Throng to the
Challenger's Waiting Room;
    say "With a groan of gears, the east wall swings
open! If you've lost now, well, you've lost..."



Test me with "e / push button / e / w".
```

Our instructions about pushing the button will be further explained in the chapter on Actions, but the thing to note here is that we can "change (whatever) exit" in order to set or re-set map directions. Notice that we have to set both directions explicitly: changing the east exit of the Waiting Room does not automatically also change the west exit of Amid the Cheering Throng.

This allows greater flexibility in our games but does require an extra line or so of work.

ExampleThe World of Charles S. Roberts

Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.

Wargaming is an ancient pursuit, but its modern form began as a professional training exercise in 19th-century Prussian staff colleges; since at least as early as H. G. Wells's "Little Wars" (1913) it has been a hobby of "boys from twelve years of age to one hundred and fifty and for that more intelligent sort of girl who likes boys' games and books." The free-form tabletop game used miniature figures and tape-measured movements, and remains the dominant form today. But in the mid-20th century, map grids on printed sheets gave the hobby a sudden new lease of life. They were easier to set up, more interesting to look at, cheaper to sell by mail-order. 1970s sales figures for "Strategy and Tactics", the leading US subscription-based wargame distributor, were very similar to those of Infocom's IF games in the 1980s. And like classical IF, the grid-based wargame parceled up a continuous world into locations.

Grids were initially square, as on a chessboard, but square cells have several disadvantages. Four directions of movement (N, E, S, W) is too few, yet allowing movement in the diagonal directions means allowing tanks to travel about 1.4 times faster northeast than they do north.

Square grids also only conform cleanly to man-made landscape features such as buildings in one orientation, and they never fit hills well. (A compromise measure to fix this, cutting the squares into octagons to leave smaller diamond squares at corner intersections, has never caught on.)

But following Charles S. Roberts's American Civil War designs for Avalon Hill of 1958-61 (notably "Chancellorsville" and the second edition of "Gettysburg"), a hexagonal grid became the new standard. Each hexagon is the same distance from the centre of all six of its neighbours, which are at equal angular spacings; and clumps of hexagons fit the shape of lakes, contoured hills, and so forth, much more naturally than clumps of squares do. Hexes also have a certain mystique - an air of "I don't belong in the children's department".

But hexes are tricky for IF, not least because English lacks words for "the direction 60 degrees around from front". Our cognitive view of the world tends to be square, perhaps because our two eyes both face front, in a direction at right angles to the plane of our arms, legs, pelvis and eyes. We reach out sideways at right angles to our walking. Even early hex-grid wargames called the cells "squares", though "hexes" eventually caught on. Still and all:

```
Forward is a direction. Forward has opposite backward.
Understand "f" as forward.
Backward is a direction. Backward has opposite
forward. Understand "b" and "back" as backward.
Forward left is a direction. Forward left has opposite
backward right. Understand "fl" as forward left.
Forward right is a direction. Forward right has opposite
backward left. Understand "fr" as forward right.
Backward left is a direction. Backward left has opposite
forward right. Understand "bl" as backward left.
Backward right is a direction. Backward right has
opposite forward left. Understand "br" as backward
right.
```

Now to forbid the use of the compass directions:

```
A direction can be hexagonal or squared-off. A direction
is usually squared-off. Forward, backward, forward left,
forward right, backward left and backward right are
hexagonal.
```

```
Before going a squared-off direction, say "In this
hexagonally-divided landscape, squared-off directions
are not allowed." instead.
```

A slight nuisance is that, with things as they are above, typing BACKWARD produces the response "Which do you mean, backward, backward left or backward right?" To avoid that silly question, we write:

```
Does the player mean going backward: it is very likely.
Does the player mean going forward: it is very likely.
```

And now a clump of 37 hexes, in six columns of six or seven rooms each. There are many ingenious ways we could put this map together automatically, but instead we will take a deep breath and write:

```
E1 is forward of E2. "Open farmland." E2 is forward of
```

E3. "The edge of woods." E3 is forward of E4. "Deep woodland." E4 is forward of E5. "Deep woodland." E5 is forward of E6. "The rear edge of woods." E6 is forward of E7. "The start of a road leading forward right." E7 is a room. "Grassland."

F1 is forward of F2. "The edge of farmland." F2 is forward of F3. "The edge of woods." F3 is forward of F4. "Clearing in woods." F4 is forward of F5. "Deep woodland." F5 is forward of F6. "A road runs backward left to forward right." F6 is a room. "The edge of grassland."

G1 is forward of G2. "Grassland." G2 is forward of G3. "The edge of farmland." G3 is forward of G4. "A copse of trees." G4 is forward of G5. "The backward edge of woodland." G5 is forward of G6. "A bend in the road, from backward left to backward right." G6 is forward of G7. "Open farmland." G7 is a room. "Open farmland."

H1 is forward of H2. "Grassland, bordered by a hedge to the right." H2 is forward of H3. "The edge of farmland, with a hedge to forward right." H3 is forward of H4. "A copse of trees." H4 is forward of H5. "Open farmland." H5 is forward of H6. "A passing place on the road, which bends forward left to forward right." H6 is a room. "Open farmland."

I1 is forward of I2. "The end of a forward road, blocked by hedges on all sides except backward." I2 is forward of I3. "A straight road runs forward to backward, with long hedges to left and right." I3 is forward of I4. "A straight road runs forward to backward, alongside a long hedge to right." I4 is forward of I5. "A straight road runs forward to backward, alongside a long hedge to right." I5 is forward of I6. "Where three roads, forward, backward left and backward right, meet. Forward right is a thick hedge." I6 is forward of I7. "Open farmland." I7 is a room. "Open farmland."

J1 is forward of J2. "Dense woodland, with a hedge to left." J2 is forward of J3. "Grassland, with a hedge to left." J3 is forward of J4. "The edge of farmland, with a hedge to left." J4 is a room. "Open farmland, with a long hedge blocking movement forward left, backward left or backward." J5 is forward of J6. "A road running forward left to backward right, alongside a hedge." J6 is a room. "Open farmland."

F1 is forward right of E2 and backward right of E1. F2 is forward right of E3 and backward right of E2. F3 is forward right of E4 and backward right of E3. F4 is forward right of E5 and backward right of E4. F5 is forward right of E6 and backward right of E5. F6 is forward right of E7 and backward right of E6.

G1 is forward right of F1. G2 is forward right of F2 and backward right of F1. G3 is forward right of F3 and backward right of F2. G4 is forward right of F4 and backward right of F3. G5 is forward right of F5 and backward right of F4. G6 is forward right of F6 and backward right of F5.

H1 is forward right of G2 and backward right of G1. H2 is forward right of G3 and backward right of G2. H3 is forward right of G4 and backward right of G3. H4 is forward right of G5 and backward right of G4. H5 is forward right of G6 and backward right of G5. H6 is forward right of G7 and backward right of G6.

I3 is forward right of H3 and backward right of H2. I4 is forward right of H4 and backward right of H3. I5 is forward right of H5 and backward right of H4. I6 is forward right of H6 and backward right of H5.

J5 is forward right of I6 and backward right of I5. J6 is forward right of I7 and backward right of I6.

And now we have a hexagonally-gridded world. Route-finding will work;

prepositional forms like "to be mapped backward left of" exist, just as they should; and in general these directions are just as good as the square ones. (The only thing which doesn't look good is the Index map, where Inform is just unable to draw a picture because it assumes a square grid. But that has no effect on play.)

The landscape is much easier to navigate with a little diagram:

```
To say legend (D - direction):
    let destination hex be the room D from the location;
    if the destination hex is nothing, say " ";
    otherwise say the destination hex.



Carry out looking:
    say "[fixed letter spacing] \ [legend forward] /[line
break][legend forward left] ---- [legend forward right][line
break] / \[line break]--< [location] >--[line break] \ /[line
break][legend backward left] ---- [legend backward right]
[line break] / [legend backward] \[variable letter spacing]
[line break]".
```

And finally:

```
The player is in I5.



Test me with "f / forward / backward left / bl / br / br / f".
```

---

101 ExampleA&E *WI*

Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

Rules about going to regions make it easy to exclude the player from a large portion of the map, even if there are many connecting paths to the region. For instance, in this story it would be annoying to have to write a rule about all four exits by which the player could reach the film set area:

"A&E"

Winding Street is a room. Winding Street is west of Duck Pond. Sloping Street is north of Winding Street, northwest of Duck Pond, west of Stately Lawn, and southwest of Stately Home. Stately Lawn is north of Duck Pond. Stately Home is north of Stately Lawn.

Film Set is a region. Duck Pond, Stately Lawn, and Stately Home are in Film Set.

Instead of going to Film Set when the player does not carry the VIP Pass: say "A burly studio guard materializes in your path, convincing you that you would prefer to be elsewhere."

The VIP Pass is in the garbage can. The garbage can is in Sloping Street.

After going to the Film Set:
    say "Success! At last you are inside the set of 'Prouder and More Prejudiced'. Next step: locating Mr Firth.";
    end the story finally.

Test me with "e / n / e / get pass / e".

Example **Further Reasons Why All Poets Are Liars**

The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

We begin with the location and its fittings, and we create a kind of value which names the different internal positions we will allow.

"Further Reasons Why All Poets Are Liars"

Nook Obscure is a room. "Above the College kitchens, which make a humming sound, less tuneable than bees, but hardly less industrious, with shrill notes of sharp command and scolding intermixed: and below Trinity's loquacious clock, who never lets the quarters, night or day, slip by him unproclaimed, and tells the hours twice over with a male and female voice. In short, the kind of rubbish room they give to a northern villager. But you get a bed and a high shelf all of your own. And you long to find some Romantic way to look out of the window."

The window, the shelf and the bed are scenery in the Nook Obscure. The shelf and the bed are supporters. The bed is enterable.

Internal position is a kind of value. The internal positions are nowhere at all, over by the window, under the shelf and near the bed.

The box is an enterable supporter in Nook Obscure. The current box position is an internal position that varies. The current box position is near the bed. "Your packing case, stamped W. WORDSWORTH (KENDAL), is [current box position]." Instead of taking the box, say "It is filled with your peerless rock collection and too heavy to lift, but could be pushed." Instead of opening the box, say "It is securely nailed shut."

We create an action, "pushing it over to", for pushing a box around on the floor of a single location. (Calling this "pushing it over to" prevents clashes with the existing "pushing it to" action, which is for pushing things from one room to another.) Almost half of the text which defines the action is concerned with the two action variables, but they make the

implementation of everything else so much easier that we end up writing less than if we hadn't used them.

```
Understand "push [box]" as a mistake ("You can push
the box to the window, the bed or the shelf.").


Understand "push [something] to [something]" as
pushing it over to. Pushing it over to is an action
applying to two things.


The pushing it over to action has an internal position
called the old position.
The pushing it over to action has an internal position
called the new position.
Setting action variables for pushing something over to
something:
    now the old position is the current box position;
    now the new position is nowhere at all;
    if the second noun is the window, now the new
position is over by the window;
    if the second noun is the bed, now the new position is
near the bed;
    if the second noun is the shelf, now the new position
is under the shelf.


Check pushing it over to:
    if the noun is not the box, say "That's not something
you can push." instead;
    if the player is on the bed, say "You can't reach from
here." instead;
    if the player is on the noun, say "Not while you are
standing on [the noun]." instead;
    if the new position is nowhere at all, say "You can
only push [the noun] to the window, the bed or the
shelf." instead;
    if the new position is the old position, say "The [noun]
is already [new position]." instead.
Carry out pushing it over to:
    now the current box position is the new position.
Report pushing it over to:
    say "With some effort, you shove [the noun] from [old
```

position] to [new position]."

Everything which remains simply provides a couple of puzzles to test this arrangement.

```
Euclid's Elements is on the shelf. Understand "euclid" or
"book" as the Elements.
Instead of taking something (called the item) which is on
the shelf:
    if the player is on the box and the current box position
is under the shelf, continue the action;
    say "You cannot reach [the item], which is up on the
shelf."


Instead of examining the window:
    say "This window opens rather unpromisingly onto
the chapel wall opposite, so even granted the moonlight
it is dark in here. Still, surely there's a poem here
somewhere?"


Instead of examining the window when the player is on
the bed:
    say "Just a blank patch of chapel wall."


Instead of examining the window when the player is on
the box:
    if the current box position is near the bed:
        say "Tantalisingly, you are not quite able to spy the
statue.";
    otherwise if the current box position is under the
shelf:
        say "All you can see is the antechapel wall, and the
dull silver gleam of the pealing organ.";
    otherwise:
        say "At last! You can just, standing on tiptoes on
the box right up at the window, make out the top of the
statue! Of such epiphanies are Poesy born. Let's see
now... oh yes...[paragraph break]And from my pillow,
looking forth by light[line break]Of moon or favouring
stars, I could behold[line break]The antechapel where
the statue stood[line break]Of Newton with his prism
```

and silent face,[line break]The marble index of a mind for ever[line break]Voyaging through strange seas of Thought, alone.";
end the story finally.

Test me with "get on bed / x window / get off / x window /get elements / get on box / x window / get elements / push box to shelf / get off / push box to shelf / get on box / get elements / x window / get off / push box to window / get on box / x window".

Example Starry Void

Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

Sometimes we may want a room to be visible from the outside in one location, but treated as a separate location when we are inside. The simplest way to do this is to make the exterior form of the object into a door object, and to describe it differently from different vantage points. (Doors in general are described more fully in the Doors section of the Things chapter.)

"Starry Void"

The Center Ring is a room.

The magician's booth is a door. "[if the player is in Center Ring]A magician's booth stands in the corner, painted dark blue with glittering gold stars.[otherwise if the magician's booth is closed]A crack of light indicates the way back out to the center ring.[otherwise]The door stands open to the outside.[end if]".

Here we've arranged for the booth to be described in the initial room description in different ways depending on where the player is when viewing it. We might like to do the same if the player takes a closer look:

```
Instead of examining the magician's booth in the Center
Ring:
    say "It is dark blue and glittering with gold stars. [if the
booth is open]The door currently stands
open[otherwise]It has been firmly shut[end if]."
```

```
Instead of examining the magician's booth in the Starry
Void:
    say "The booth door is [if the magician's booth is
open]wide open[otherwise]shut, admitting only a thin
crack of light[end if]."
```

And now we put it in place:

```
The magician's booth is inside from Center Ring and
outside from Starry Void.
```

...and make sure that the booth-and-door object responds to all the names we have used for it in different places:

```
Understand "door" or "of" or "the" or "light" or "crack" or
"thin crack" as the booth.
```

```
Test me with "examine booth / open door of the booth /
in / examine door / close door / look / examine crack of
light".
```

A final nice touch, if we're so inclined, is to borrow from the Basic Actions chapter and make the player automatically open the booth door before trying to enter:

```
Before going through the closed magician's booth:
    say "(first opening the door of the booth)[command
clarification break]";
    silently try opening the booth.
```

For the contrasting case of a space that is nested inside another place and is not its own room -- say a stall at an open-air market, or a rowboat on a lake -- see the example "Tamed".

Example Waterworld

A backdrop which the player can examine, but cannot interact with in any other way.

It's tempting to handle the player's inability to interact with something with a simple instead rule:

```
"Waterworld 1"

A view is a kind of backdrop. Instead of doing something
other than examining to a view, say "You are too far
from [the noun] to do anything but look."

The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."

The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.

Test me with "x sun / get sun / n / x sun / n / x sun".
```

Unfortunately, the rule does not address the case where the object in question is the second noun; so for instance the following example reveals the difficulty:

```
"Waterworld 2"
```

```
A view is a kind of backdrop. Instead of doing something
other than examining to a view, say "You are too far
from [the noun] to do anything but look."


The player carries a rope.


The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."


The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.


Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to the sun".
```

...where the response here behaves as though the sun is in reach. If we had a fully implemented tying action, the player would (even more disastrously) be allowed to lasso celestial objects.

We could add a second instead rule as well:

```
"Waterworld 3"


A view is a kind of backdrop.


Instead of doing something other than examining when
the noun is a view:
    say "You are too far from [the noun] to do anything
but look."
Instead of doing something other than examining when
the second noun is a view:
```

```
    say "You are too far from [the second noun] to do
anything but look."


The player carries a rope.


The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."


The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.


Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to sun".
```

This produces acceptable output again, but there is a more elegant way, one that works better with Inform's existing world model. Currently the default model assumes that accessibility -- whether the player can reach something or not -- is checked between the Before... rules and the Instead... rules. We can add our own accessibility rules, including this one to govern whether views are accessible. So for instance:

```
"Waterworld 4"


A view is a kind of backdrop.


The can't touch views rule is listed before the access
through barriers rule in the accessibility rulebook.


Accessibility rule (this is the can't touch views rule):
    if the action requires a touchable noun and the noun
```

```
is a view:
        say "You are too far from [the noun] to do anything
but look." instead;
    if the action requires a touchable second noun and
the second noun is a view:
        say "You are too far from [the second noun] to do
anything but look." instead;



The player carries a rope.



The sun is a view. It is everywhere. The description is
"A blazing sun makes you wish you had never been
born."



The Sahara is a room. North of the Sahara is More
Sahara. North of More Sahara is Yet Further Sahara.



Test me with "x sun / get sun / n / x sun / n / x sun / tie
rope to sun".
```

Now our new accessibility rule fits into its proper stage.

Example **Tiny Garden** *WI*

A lawn made up of several rooms, with part of the description written automatically.

Sometimes we want to make a list of something too complicated to express in a say list... phrase. When this happens, we can instead mark all the items we want to mention as "marked for listing".

In this case, we have a lawn area made up of four rooms. We want each room to automatically describe the directions leading to the other parts of the lawn. To do this, we will first determine which directions are relevant and mark those for listing, then list them.

"Tiny Garden"

The Herb Garden is a room. "Along this side of the house run your great-aunt's herb beds."

A Grassy Room is a kind of room. The printed name of a Grassy Room is usually "Lawn". The description of a Grassy Room is "The grass underfoot is thick and green. The lawn extends to [grassy directions] from here."

The following phrase goes through all the directions in the compass and marks the ones that are interesting to us at the moment.

```
To say grassy directions:
    repeat with that way running through directions:
        if the room that way from the location is a grassy
room,
            now that way is marked for listing;
    say "[a list of directions which are marked for listing]";
    now every direction is not marked for listing.
```

Lawn1 is west of the Herb Garden. It contains a picnic table and a wicker basket. Lawn2 is south of Lawn1 and southeast of Lawn4. Lawn3 is southwest of Lawn1, west of Lawn2, and south of Lawn4. Lawn4 is west of Lawn1. Lawn4 contains a birdbath. The birdbath is fixed in place.

Lawn1, Lawn2, Lawn3, and Lawn4 are Grassy Rooms.

Test me with "w / s / w / n".

Signposts such as those provided on hiking paths in the Swiss Alps, which show the correct direction and hiking time to all other locations.

The following rule appends a paragraph to every room description. We need not worry about doors (despite the pass in the Bernese Oberland known figuratively as the "Little Door").

```
"Hotel Stechelberg"

After looking:
    say "Yellow arms on the signpost point:-[line break]";
    repeat with destination running through interesting
rooms:
        let the way be the best route from the location to
the destination;
        if the way is a direction, say " [way] for [the
destination]: [number of moves from the location to the
destination] Std."

Hotel Stechelberg is a room. "The wooden hiking inn at
the end of the road, with flowerboxes, canton flags,
outdoor tables and a triangular paddock for the cows
contesting the annual Miss Stechelberg competition.
Otto and Marianne do cheerful innkeeper things, while
the sun blazes from a gentian-blue sky."

A room can be dull or interesting. A room is usually dull.

North of Hotel Stechelberg is Trummelbachfalle. North
of Trummelbachfalle is Lauterbrunnen. Lauterbrunnen is
interesting.

Southeast of Hotel Stechelberg is Trachsellauenen.
Trachsellauenen is interesting.
```

```
Test me with "look".
```

With a bit more work, the result might be:

```
Hotel Stechelberg
The wooden hiking inn at the end of the road, with
flowerboxes, canton flags, outdoor tables and a
triangular paddock for the cows contesting the annual
Miss Stechelberg competition. Otto and Marianne do
cheerful innkeeper things, while the sun blazes from a
gentian-blue sky.


Yellow arms on the signpost point:-
    north for Lauterbrunnen: 2 Std.
    west for Sefinental: 2 Std.
    west for Schilthorn: 6 Std.
    southeast for Trachsellauenen: 1 Std.
    southeast for Oberhornsee: 3 Std.
```

ExampleCarnivale

An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

Suppose we want to make an object that (unlike a backdrop) is definitely located in one room, but can be seen from far off. We want to allow the player to interact with it from a distance, but only using those actions that require visibility. Other actions should be denied:

```
"Carnivale"


The Fairground is a region. Park Entrance, By the
Wheel, and Candy Stand are in Fairground. Candy
Stand is north of By the Wheel. Park Entrance is west of
Candy Stand and northwest of By the Wheel.
```

```
The ferris wheel is scenery in By the Wheel. "It is
extravagantly tall and carries several dozen glass
gondolas for riders."
```

```
The description of By the Wheel is "You stand at the
foot of an enormous ferris wheel, which turns far too
quickly and never seems to stop for new riders."
```

```
The description of Park Entrance is "You are now just
inside the gates. Behind you snakes a triple line of
fairgoers all the way down the length of the valley to the
railway station. Roughly southeast of here is the ferris
wheel, towering over the other attractions."
```

```
The description of Candy Stand is "A hut in pale pink
and baby blue dispenses marshmallow death's-heads,
sugar-beetles, and other such treats. The giant ferris
wheel is just off to the south from here."
```

As the descriptions make clear, the ferris wheel should be visible from everywhere in the fair, so we'll borrow a line from the Activities chapter to make that happen:

```
After deciding the scope of the player:
    if the location is in Fairground, place the ferris wheel
in scope.
```

"Scope" determines what the player can interact with; by writing this rule, we make Inform understand all commands that refer to the ferris wheel when the player is anywhere in the fairground, instead of responding with

```
You can't see any such thing.
```

as it normally would.

Now, by default, if the player were to type TOUCH FERRIS WHEEL

while in another room, he would get the response

```
You can't reach into By the Wheel.
```

This may not be quite what we want, but we can replace this text with our own reaching inside rule:

```
Rule for reaching inside a room:
    say "You can only look from this distance.";
    deny access.
```

And because our accessibility rules are considered before the "Instead" phase, we can write the following rule confident that it will apply only when the player is in fact in range to touch the ferris wheel:

```
Instead of touching the ferris wheel:
    say "You don't dare: it's spinning too fast."
```

```
Test me with "x ferris wheel / touch ferris wheel / se / x
ferris wheel / touch ferris wheel".
```

ExampleEddystone

Creating new commands involving the standard compass directions.

Using the compass directions in commands is a little bit finicky because directions are forbidden to figure in any interactions involving touch. (Really, directions are more a concept than an object; this is a compromise situation.) In any case, if we want to write a new command involving these, we need to be sure to specify that the direction is a visible thing. For instance:

```
"Eddystone"
```

```
The Lighthouse is a room. "A lonely place, but in these
tense times, no one but the lighthouse keeper and a few
trusted agents are allowed on the grounds at all, for fear
of sabotage."


The light is a fixed in place thing in the Lighthouse. "At
the center of the room is the light itself, a 1000-Watt
tungsten halogen light powered by diesel generator, and
having a visible range of twenty-six nautical miles."
Understand "lamp" as the light. It is lit. The light has a a
direction called heading. The heading of the light is
north.


A room is usually dark.


Understand "turn [something] [a direction]" as
reorienting it to. Reorienting it to is an action applying to
two things.
```

will give us

```
>turn light northeast
You must name something more substantial.
```

To avoid this mystifying result:

```
"Eddystone"


The Lighthouse is a room. "A lonely place, but in these
tense times, no one but the lighthouse keeper and a few
trusted agents are allowed on the grounds at all, for fear
of sabotage."
```

The light is a fixed in place thing in the Lighthouse. "At the center of the room is the light itself, a 1000-Watt tungsten halogen light powered by diesel generator, and having a visible range of twenty-six nautical miles." Understand "lamp" as the light. It is lit. The light has a direction called heading. The heading of the light is north.

A room is usually dark.

Understand "turn [something] [a direction]" as reorienting it to. Reorienting it to is an action applying to one thing and one visible thing.

Instead of turning the light, say "Try turning the light to the direction of your choice."

Check reorienting it to: if the noun is not the light, say "You couldn't do so meaningfully." instead; if the second noun is up or the second noun is down, say "The light only points in compass directions." instead.

And now that's done, we have a little fun calculating where the beam hits:

```
Carry out reorienting it to:
    now the heading of the light is the second noun;
    let way be the heading of the light;
    let place be the room way from the Lighthouse;
    while place is a room and place is lower than
Lighthouse:
        let place be the room way from the place;
    if place is not a room, now the beam is nowhere;
    otherwise move beam to the place.
```

Report reorienting it to: say "The light now points [heading of the light][if the beam is in a room], spotlighting [the holder of the beam][otherwise], into

empty space[end if]."

The beam is a lit thing. Understand "light" or "brilliant" as the beam. "Brilliant light from the lighthouse floods the whole area." It is fixed in place. Instead of doing something other than examining to the beam: say "The light is, of course, intangible." The description is "The light is coming from the lighthouse, since the lamp is apparently pointed this way."

Altitude is a kind of value. 200 ft specifies an altitude. A room has an altitude. The altitude of a room is usually 50 ft. The altitude of the Lighthouse is 100 ft.

Definition: a room is low if its altitude is 20 ft or less.

The Jetty is south of the Lighthouse. "During daylight hours, a fine place to catch almost unlimited supplies of crayfish. Less entertaining by night."

North of the Lighthouse is the Uphill Road. The altitude of Uphill Road is 75 ft. North of Uphill Road is Hilltop. The altitude of Hilltop is 110 ft. The description of Hilltop is "The highest natural point around for miles; sometimes you will sit up here and watch for the lighthouse supply ship, the Lady Loch."

Northeast of the Lighthouse is Open Field. East of the Lighthouse is Stanley Creek Valley. The description of Stanley Creek is "This place used to have some other name meaning Ghost Valley in the aboriginal language, but it was piously renamed by missionaries." Train Trestle is east of the Stanley Creek Valley. "The now-abandoned track of the Bush Pacific Railway runs here, above Stanley Creek." The altitude of Train Trestle is 100 ft.

```
Before going from a room (called source) to a room
(called destination):
    if source is lower than destination:
        say "It's an uphill climb...";
    otherwise:
        if destination is lower than source, say "You're
heading downhill now...";
        otherwise say "It's a straight shot."



Test me with "turn lamp / turn lamp down / turn lamp
east / e / e".
```

All very loosely based on the Eddystone Point lighthouse of Tasmania, built in 1889, and forbiddingly remote even today. George Isaacs, a child growing up in the lighthouse, remembers the plentiful crayfish.

---

366 **ExampleRock Garden** *WI*

A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

A map of linked rooms works well for modeling enclosed or indoor space, and somewhat less well for modeling large open spaces, where a person should reasonably be able to see things which are much too far away to touch. With some modifications to scoping, though, we can create an environment where objects in nearby rooms are described and viewable, and where the player will automatically move towards distant items before interacting with them physically.

```
"Rock Garden"



Section 1 - General Rules



Intervisibility relates rooms to each other in groups. The
verb to be connected with means the intervisibility
```

relation.

Definition: a room is inter-visible if it is connected with more than one room.

After deciding the scope of the player when the location is an inter-visible room:
    repeat with other place running through rooms which are connected with the location:
        unless the other place is the location, place the other place in scope.

Rule for reaching inside a room (called target) which is connected with the location:
    let way be the best route from the location to the target;
    if the way is not a direction:
        say "You can't get over to [the target] from here.";
        deny access;
    say "(first heading [way])[command clarification break]";
    try going way;
    if the player is in the target, allow access;
    otherwise deny access.

After looking when the location is an inter-visible room:
    repeat with other place running through rooms which are connected with the location:
        if the other place is not the location, describe locale for other place.

Section 2 - The Scenario

Rock Garden West is west of Rock Garden East. Rock Garden East contains a rake. Rock Garden West contains a bench and a maple leaf. The bench is an

enterable supporter.

Rock Garden West is connected with Rock Garden East.

Test me with "get rake / drop rake / sit on bench / get rake".

Example **A View of Green Hills**

WI

A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

Suppose a game in which the player is wandering an open landscape with long vistas, allowing him to LOOK in some direction, or even look at an adjacent location.

"A View of Green Hills"

Corinth is a room. Athens is east of Corinth. Epidaurus is southeast of Corinth and east of Mycenae. Mycenae is south of Corinth. Olympia is west of Mycenae. Argos is south of Mycenae. Thebes is northwest of Athens. Pylos is south of Olympia. Sparta is east of Pylos and south of Argos. Delphi is northwest of Thebes.

Understand "look [direction]" as facing.

Facing is an action applying to one visible thing.

```
Carry out facing:
    let the viewed item be the room noun from the
location;
    if the viewed item is not a room, say "You can't see
anything promising that way." instead;
    try looking toward the viewed item.
```

In rules about action handling, "noun" refers to the first object that the player has mentioned in his command, so if the player typed >LOOK WEST, "let the viewed item be the room noun from the location" would be processed as "let the viewed item be the room west from the location", and so on.

We can at need override the default behavior, if it is not going to be appropriate for the player to see the next room over. There is only sky above at any time, so...

```
Instead of facing up:
    say "Above you is bright sky."
```

```
Understand "look toward [any adjacent room]" as
looking toward. Understand "examine [any adjacent
room]" as looking toward.
```

```
Looking toward is an action applying to one visible thing.
```

```
Carry out looking toward:
    say "You make out [the noun] that way."
```

This design allows us to create descriptions for rooms (as seen from the outside) which will work regardless of where we're looking from. For instance:

```
Instead of looking toward Athens:
    say "Even from here you can make out the silhouette
of the Acropolis."
```

```
Test me with "look north / look south / look up / look
east / east / look west".
```

Example Stately Gardens *WI*

An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

This time we're going to assume that the player can see into any room that is on a line of sight within one or two steps of travel.

```
"Stately Gardens"


Chapter 1 - Laying Out Rooms


A room can be indoors or outdoors.


Use full-length room descriptions.


After deciding the scope of the player:
    repeat with the way running through directions:
        let first step be the room the way from the location;
        if the first step is a room:
            place the first step in scope;
            let second step be the room the way from the
first step;
            if the second step is a room, place the second
step in scope;
    place the obelisk in scope.
```

The obelisk is so large that it can be seen from every room. If we had a number of such large monuments we might want to write a systematic routine to handle them, but this will do for now.

```
The room description heading rule is not listed in the
carry out looking rules.
```

Now, we set things up so that the surrounding areas are described automatically as part of the room description:

```
Building description is a truth state that varies. Building
description is false.


After looking when the location is an outdoors room:
    now count of sentences is 0;
    now building description is true;
    repeat with way running through directions:
        let space be the room way from the location;
        if space is an outdoors room, silently try looking
toward space;
    if the obelisk is not in the location and the obelisk is
unmentioned:
        let the way be the best route from location to the
Upper Terrace;
        if the way is a direction, say "[The obelisk] is
proudly visible on [the way] horizon. [run paragraph
on]";
        increment the count of sentences;
    now building description is false;
    unless the count of sentences is 0:
        say paragraph break.
```

But perhaps there are a few rooms where we do not wish that to happen, so we'll build in exceptions for those.

```
After looking in the rose garden:
    say "Otherwise, you are quite cut off.".


After looking in the Ha-ha:
    do nothing.
```

And suppose we want to allow the player to look in any direction:

```
Understand "look [direction]" or "look to/toward
[direction]" as facing.


Facing is an action applying to one visible thing.


Carry out facing:
    let the viewed item be the room noun from the
location;
    if the viewed item is not a room:
        if the location is indoors, say "Your view is
restricted by the lack of doors or windows in that
direction." instead;
        otherwise say "You can't see anything promising
that way." instead;
    try looking toward the viewed item.


Instead of facing up:
    say "Above you is bright sky."
```

We also need to tell distant rooms how to describe themselves.

```
Understand "look toward [any adjacent room]" as
looking toward.


Looking toward is an action applying to one visible thing.


Check looking toward a room which does not contain
something mentionable:
    if building description is false:
        say "You can't make out anything of interest that
way." instead.
```

```
Carry out looking toward:
    now every thing is unmentioned;
    now the chosen direction is the best route from the
location to the noun;
    now the second noun is the room the chosen
direction from the noun;
    if the noun contains something mentionable:
        repeat with item running through mentionable
things in the noun:
            carry out the writing a distant paragraph about
activity with the item;
    if the noun contains something mentionable:
        increment the count of sentences;
        choose row count of sentences in the Table of
Distance Sentences;
        if the second noun is an outdoors room and the
second noun contains something mentionable, say
"[both entry] [run paragraph on]";
        otherwise say "[here entry] [run paragraph on]";
    otherwise:
        if the second noun is an outdoors room and the
second noun contains something mentionable:
            increment the count of sentences;
            choose row count of sentences in the Table of
Distance Sentences;
            say "[there entry] [run paragraph on]";
    if building description is false:
        say paragraph break.
```

And again, some exception needs to be made for seeing what's in the dip in the ground:

```
Instead of looking toward the Ha-ha:
    now the chosen direction is the best route from the
location to the noun;
    now the second noun is the room the chosen
direction from the noun;
    if the second noun is an outdoors room and the
second noun contains something mentionable:
        increment the count of sentences;
        choose row count of sentences in the Table of
Distance Sentences;
        say "[there entry] [run paragraph on]".
```

The following is to account for cases where the player types "look toward obelisk" or similar, rather than looking toward a room:

Understand "look toward [something]" as examining.

The following is arguably an unnecessary refinement, but the listing of items in the distance gets a bit repetitive unless we vary the sentence structure.

Chosen direction is a direction that varies.

Count of sentences is a number that varies.

Table of Distance Sentences

here	"From here, you make out [a list of mentionable things in the noun]."
a little way [chosen direction]	"To the [chosen direction], [a list of mentionable things in the second noun]."
a little way [chosen direction]	"To the [chosen direction] there [is are a list of mentionable things in the second noun], partly obscuring your further view of [a list of mentionable things in the second noun]."
a little way [chosen direction]	"Meanwhile, to the [chosen direction] is the [a list of mentionable things in the second noun]."
a little way [chosen direction]	"When you look [chosen direction], you see [a list of mentionable things in the noun], and so on further on [a list of mentionable things in the second noun]."
a little way [chosen direction]	"Somewhere [chosen direction] is a [a list of mentionable things in the second noun]."
a little way [chosen direction]	"To the [chosen direction] [a list of mentionable things in the second noun]."
a little way [chosen direction]	"Meanwhile, to the [chosen direction] is the [a list of mentionable things in the second noun]."
a little way [chosen direction]	"Finally, to the [chosen direction] [a list of mentionable things in the second noun]."

Now, our ability to view things at a distance should be determined by the size of the things we're trying to see:

Chapter 2 - Height

A height is a kind of value. 10 feet 11 inches specifies a height. 10 feet 11 specifies a height. The verb to stand means the height property. The verb to measure means the height property. A thing has a height. The height of a thing is usually 3 feet 0.

Definition: a thing is tiny if its height is 0 feet 6 inches or less.

Definition: a thing is short if its height is 3 feet 0 or less.

Definition: a thing is tall if its height is 6 feet 0 or more.

The height of a man is usually 5 feet 10 inches. The height of a woman is usually 5 feet 6 inches.

Definition: a thing is monumental if it is taller than 25 feet 0 inches.

Definition: a thing is mentionable if it stands tall enough to see.

To decide whether (item - a thing) stands tall enough to see:
    if the item is in the Rose Garden and the item is shorter than the roses, no;
    if the item is mentioned, no;
    if the item is in an adjacent room and item is taller than 2 feet 0, yes;
    if the item is taller than 4 feet 0, yes;

```
    no.


Instead of examining something which is within a room
(called the space) which is not the location:
    if the location is adjacent to the space:
        if the noun is tiny, say "It is too far from here for
you to make out much detail about [the noun]." instead;
        let way be the best route from the location to the
space;
        if the way is a direction, say "You gaze off [way] at
[the noun]...";
        continue the action;
    otherwise:
        if the noun is short, say "It is too far from here for
you to make out much detail about [the noun]." instead;
        let way be the best route from the location to the
space;
        if the way is a direction, say "You gaze off [way]
into the distance at [the noun]...";
        continue the action.
```

We might also want to be able to override, manually, the way distant things are described.

```
Writing a distant paragraph about something is an
activity.


Rule for writing a distant paragraph about the lily pond:
    if the second noun is a room and something
mentionable is in the second noun, say "A [lily pond],
[chosen direction], patchily reflects [a list of mentionable
things in the second noun] on the far side. [run
paragraph on]";
    otherwise say "To [the chosen direction], [a lily pond]
shimmers in the sunlight. [run paragraph on]"


Rule for writing a distant paragraph about the roses:
    if something in the Rose Garden is taller than the
roses,
        say "Over the tops of [the roses], [chosen
direction], you see [a list of mentionable things in the
```

```
rose garden]. [run paragraph on]";
    otherwise say "Immediately [chosen direction] is [the
roses]. [run paragraph on]"
```

```
Rule for writing a distant paragraph about the obelisk:
    if a mentionable thing in the Upper Terrace is shorter
than the obelisk,
        say "A stupidly grand [obelisk], [chosen direction],
towers over [a list of mentionable things in the Upper
Terrace]. [run paragraph on]";
    otherwise say "To [the chosen direction], you can't
help noticing [the obelisk], which is much larger than
any object really needs to be. [run paragraph on]".
```

```
After writing a distant paragraph about something:
    increment the count of sentences.
```

Moreover, proximate things might have special descriptions too.

```
Rule for writing a paragraph about something tiny when
the location is outdoors:
    if the location is the Gravel Circle,
        say "Abandoned in the gravel [is-are a list of
unmentioned tiny things in the location]. [run paragraph
on]";
    otherwise say "Half trampled into the grass, and easy
to miss, [is-are a list of unmentioned tiny things in the
location]. [run paragraph on]"
```

```
Before doing something other than examining or
approaching to something which is not within the
location:
    if the player has the noun, continue the action;
    say "(first going over to [the noun])[line break]";
    try approaching the noun;
    if the noun is not within the location, stop the action.
```

```
Understand "go toward/to/towards/near [something]" or
"approach [something]" as approaching.
```

```
Approaching is an action applying to one visible thing.


Check approaching:
    if the player is in something, say "You'll have to get
up." instead;
    if the noun is within the location, say "You're as close
to [the noun] as you can get." instead;
    let space be the location of the noun;
    if the space is not a room, say "You don't quite see
how to get there." instead;
    let way be the best route from the location to the
space;
    unless way is a direction,
        say "You can't see how to get over there from
here." instead.


To head to (space - a room):
    let the way be the best route from the location to the
space;
    if the space is adjacent to the location,
        try going way;
    otherwise silently try going way.


Carry out approaching:
    let space be the location of the noun;
    while the space is not the location:
        head to space.
```

This is a bit primitive, since if we had an occasion where going was blocked, we could get stuck in a loop. So we would need to be careful, but for this example it won't arise.

```
Going state is a truth state that varies. Going state is
false.


Check going:
```

now going state is true.

The description of a room is usually "[if going state is true]You drift [noun] across the open lawn[direction relative to obelisk]. [end if]An absolutely phenomenal quantity of manicured turf stretches from where you stand in almost every direction."

Before reading a command:
now going state is false.

To say direction relative to obelisk:
if obelisk is in the location:
say ", as though drawn magnetically to the foot of the monument";
otherwise:
let way be the best route from the location to the Upper Terrace;
if way is the noun, say ", drawn towards [the obelisk]";
if the way is the opposite of the noun, say ", keeping [the obelisk] more or less at your back".

Chapter 3 - The Grounds

When play begins:
now the left hand status line is "Idyllic";
now the right hand status line is " ".

The Gravel Circle, the Ha-ha, the Sheep Field, the Open Lawn, the Croquet Ground, the Rose Garden, the Upper Terrace, the Middle Terrace, and the Lower Terrace are outdoors.

The Middle Terrace is north of the Lower Terrace and south of the Upper Terrace. The lily pond is fixed in place in the Middle Terrace."You [if going state is true]come to[otherwise]are at[end if] the north edge of a perfectly round lily pond, bordered with stones. Its surface patchily reflects [the marble anteater] on the south bank." A tent peg and a wilted orchid are in the Middle Terrace. The tent peg measures 0 feet 6. The orchid measures 0 feet 4.

The description of the Lower Terrace is "[if going state is true]You climb [noun] up a small hillock[direction relative to obelisk][otherwise]You stand on a short, round, entirely artificial hillock[end if]."

The marble anteater is a fixed in place thing in the Lower Terrace. The height of marble anteater is 6 feet 2 inches."A marble anteater stands on a pedestal at the top of the hill. In the bright sunlight the white marble makes a striking contrast with [the obelisk] in the distance." The description is "The anteater is very much more than life-size."

The obelisk of black granite is a fixed in place thing in the Upper Terrace."Now that you are at the foot of it, you can properly appreciate the stupid immensity of the obelisk, pointing stonily at heaven." The height of the obelisk is 50 feet 0 inches. The description of the obelisk is "It stands ridiculously tall, and has an inscription on the face."

The inscription is part of the obelisk. The height of the inscription is 0 feet 3 inches. The description of the inscription is "You can't read the squirming, pointed letters, but they make you uneasy.".

The Gravel Circle is west of the Upper Terrace, northwest of the Middle Terrace, and north of the Croquet Ground. The description of the Gravel Circle is

"[if going state is true]You head [noun] until the lawn thins and[otherwise]Here the lawn[end if] gives way to a circle of raked gravel, which crunches pleasingly beneath you."

Instead of going northwest in the Upper Terrace, try going north.

The Ha-ha is north of the Gravel Circle and northwest of the Upper Terrace. The description of the Ha-ha is "[if going state is true]The land dips here so suddenly that you do not know the dip is there until you're in it; but it prevents livestock from crossing barriers, and that is the important thing[otherwise]You are at the base of a steep-sided depression, so the lawn continues north and south more or less at the level of your head[end if].

The tip of [the obelisk] is the only thing you can make out from this depression, off to the southeast.". North of the Ha-ha is the Sheep Field. In the Sheep Field is an animal called a black sheep. The black sheep stands 4 feet 3 inches."A black sheep grazes placidly nearby." The description of the black sheep is "It reminds you of your Uncle Tim."

Before going from the Ha-ha:
    say "It's a bit of a scramble to get back up the side of the depression, and you keep slipping in the damp grass. But you manage at last."

The Rose Garden is southwest of the Lower Terrace. The thicket of red roses is a fixed in place thing in the Rose Garden. The thicket stands 4 feet 2 inches."Heavy red roses grow over a roughly horseshoe-shaped wall around you. Over this barrier, the head of [the marble anteater] is visible to the northwest, and the tip of [the obelisk] in the distance."

The description of the Rose Garden is "[if going state is true]You slip [noun] into the enclosure of the rose garden. [end if]The rest of the park, and the world, seems muted and quiet."

Instead of smelling the rose garden: try smelling the roses. Instead of smelling the roses, say "The smell tickles the back of your throat and makes you want to cough."

Instead of listening to the rose garden:
    say "You can't hear anything at all."

The Open Lawn is north of the Rose Garden, west of the Lower Terrace, and southwest of the Middle Terrace. The Croquet Ground is north of the Open Lawn, west of the Middle Terrace, southwest of the Upper Terrace, and northwest of the Lower Terrace.

A discarded champagne cork is in the Open Lawn. It stands 0 feet 2 inches.

A stone bench is an enterable supporter in the Croquet Ground. It stands 3 feet 8 inches."There is a stone bench here -- a sort of stone sofa, really, with nymphs disporting themselves on the arms and back." The description of the bench is "It used to be a Roman sarcophagus -- hence the nymphs -- but someone has thoughtfully recarved it as lawn furniture."

The half-size Bentley is a vehicle in the Gravel Circle."A sort of child's-toy version of a Bentley is parked [if something parkable is in the location]beside [the tallest parkable thing in the location][otherwise]close at hand[end if]." The description of the half-size Bentley is "Of beautiful and unambiguously luxurious lines, but

sized down to hold only one or (at a stretch) two people, and powered by electricity." The half-size Bentley stands 3 feet 6 inches.

Definition: a thing is parkable if it is not a person and it is not the Bentley.

Instead of touching the obelisk, say "Though it is black stone in sunlight, the obelisk is very cold to the touch."

Test me with "look east / look toward obelisk / s / s / e / sw / ne / n / n / w / n / n / examine obelisk / touch obelisk / read inscription".

---

45 Example Something Narsty *WI*

A staircase always open and never openable.

In a game with many staircases, we might want:

A staircase is a kind of door. A staircase is always open.
A staircase is never openable.

Defining the staircase this way means that we will never be able to get away with (for instance) a folding ladder into the attic which is sometimes closed up. So alternatively we might do

"Something Narsty"

A staircase is a kind of door. A staircase is usually open.
A staircase is seldom openable.

We could then write a rule so that whenever the player types CLIMB [any staircase], the command is diverted to an enter command, while all other doors refuse to respond to such treatment. Still, this kind is now usable, as we see in this trivial example:

```
The ladder is a staircase. It is above the Woodshed and
below the Scary Loft.

Test me with "up".
```

But these are refinements for a later chapter.

---

63 **Example When?** *WI*
A door whose description says "...leads east" in one place and "...leads west" in the other.

Very simple, but quite frequently useful:

```
"When?"

The temporal vortex is an open door. It is west of
Yesterday and east of Today. "A whirling temporal
vortex leads [if the player is in
Yesterday]west[otherwise]east[end if]."
```

---

89 **Example Hayseed** *WI*
A refinement of our staircase kind which can be climbed.

Presumably all staircase-type connections between rooms should respond when the player says CLIMB STAIRS (or the equivalent). So

```
"Hayseed"

A staircase is a kind of door. A staircase is usually open.
A staircase is seldom openable.

The ladder is a staircase. It is above the Barn and below
the Hayloft.

Instead of climbing a staircase:
    try entering the noun.

Test me with "climb ladder / g".
```

Attempts to climb other types of door will still be treated as useless.

Example **Higher Calling** *WI*

All doors in the game automatically attempt to open if the player approaches them when they are closed.

"Called" is quite useful in the context of rules about going, since go rules often refer to things other than the noun the player typed. For instance, suppose we want to have doors which politely open when the player tries to pass through them:

```
"Higher Calling"

Before going through a closed door (called the blocking
door):
    say "(first opening [the blocking door])[line break]";
```

```
silently try opening the blocking door;
if the blocking door is closed, stop the action.
```

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.
```

```
The church door is east of Eastern End and west of the
Courtyard. The church door is a door.
```

```
Test me with "e / e".
```

A fuller set of such rules is included in the Locksmith extension.

Example: Wainwright Acts

A technical note about checking the location of door objects when characters other than the player are interacting with them.

Suppose we wanted to write rules for a character who will interact with doors in other locations even when the player is not present. This poses a little challenge: doors are actually single objects, and -- with the same shuffling of stage properties that applies to backdrops -- they are moved as needed to represent the door object in whatever room contains the player.

That means that it isn't safe to rely on a phrase like

```
if an open door is in the location of Bernard
```

because, even if Bernard's location is connected by doors to other places, the actual representation of that door may not be "in" Bernard's location, from the model's point of view, at this exact moment.

This does not, of course, mean that we can't ask this question; just that we have to be a little cleverer about how we phrase it. Every door has properties that correspond to the two locations

linked:

```
the front side of the blue door (a room, which is
arbitrarily one side of the door)
the back side of the blue door (arbitrarily the other side)
```

We can make this information easier to check with a conditional relation, like so:

```
Liminality relates a door (called X) to a room (called Y)
when the front side of X is Y or the back side of X is Y.
The verb to be a threshold of means the liminality
relation.
```

And this allows us to write rules that have characters interacting with doors even in the player's absence:

```
"Wainwright Acts"


The Waiting Room is a room. The waiting room door is
west of the Waiting Room and east of the Gents' Loo.
The Waiting Room door is an open door. "The waiting
room door [if open]stands open[otherwise]is shut
firmly[end if]."


Sir Humphrey is a man in the Gents' Loo.


Liminality relates a door (called X) to a room (called Y)
when the front side of X is Y or the back side of X is Y.
```

The verb to be a threshold of means the liminality relation.

Definition: a person is other if he is not the player.

```
Every turn:
    repeat with indiscreet one running through other people:
        repeat with port running through open doors that are a threshold of the location of the indiscreet one:
            if the port is a threshold of the location and the indiscreet one is not in the location:
                say "Through [the port], you overhear [the indiscreet one] discussing [one of]his hopes for your imminent resignation[or]your wife's infidelity[or]your financially straitened circumstances[or]ways to avoid attending your birthday party[or]your halitosis[as decreasingly likely outcomes]."
```

Test me with "z / z / z / w / z / e / close door / z".

Example **Whither?**

WI

A door whose description says where it leads; and which automatically understands references such as "the west door" and "the east door" depending on which direction it leads from the location.

Here we expand on the simple examples When? and Whence?; this time we want the player to be able to refer to doors by their directions, as in "the west door" when the door in question does in fact lead west.

"Whither?"

The temporal vortex is an open door. It is west of Yesterday and east of Today.

```
The initial appearance of a door is usually "Nearby [an
item described] leads [if the other side of the item
described is visited][direction of the item described from
the location] to [the other side][otherwise][direction of
the item described from the location][end if]."


Direction-relevance relates a door (called X) to a
direction (called Y) when the direction of X from the
location is Y. The verb to be directionally-relevant to
means the direction-relevance relation.


Understand "[something related by direction-relevance]
door" as a door.
```

As an added touch, we respond also to the case where the player postulates a door in some direction when there is no such thing at the moment:

```
Rule for printing a parser error when the player's
command includes "[non-door direction] door":
    say "There is no door in that direction." instead.


Definition: a direction (called direction D) is non-door:
    let the target be the room-or-door direction D from the
location;
    if the target is a door:
        no;
    yes;


Test me with "examine west door / x east door / w / x w
door / x e door / tie me to the west door / tie the west
door to me / push the west door east / push the east
door west".
```

Example **Neighborhood Watch**

A locked door that can be locked or unlocked without a key from one side, but not from the other.

Suppose we want a locked door that can be opened with a key, but is also openable by hand without a key from one side only. We start by defining an ordinary lockable door and the key that controls it:

```
"Neighborhood Watch"

The shabby door is a door. It is outside from the Studio
Apartment and inside from the Rickety Stairwell. The
shabby door is locked.

The brass key is carried by the player. It unlocks the
shabby door.
```

The next part is going to require that we modify the normal operation of the "lock" command. "Lock" ordinarily requires that the player supply two objects: a thing he wants to unlock, and the key he wants to use on it. The full command is LOCK DOOR WITH THE KEY, and Inform will not accept simply LOCK DOOR as locking.

Therefore, we're going to need to create our own new variant on the lock verb (and the unlock verb, while we're at it). The full procedure for this is laid out in the chapters on Action and Understanding, but here is an example:

```
Understand "lock [something]" as locking keylessly.
Locking keylessly is an action applying to one thing.
```

Here we've created a new action -- locking something without a key -- and we've told Inform to understand LOCK DOOR as this action, rather than an incomplete command to LOCK DOOR WITH SOMETHING.

Now we add some instructions so that the game will not let us use this keyless unlocking command unless we're in the right place or are properly equipped:

```
Check locking keylessly:
    if the noun is not a door, say "[The noun] is not
something you can lock." instead;
    if the noun is locked, say "[The noun] is already
locked." instead;
    if the player carries the brass key and the player is in
the Stairwell, try locking the noun with the brass key
instead;
    if the player is in the Stairwell, say "You can't lock the
door from this side without the key." instead.
```

This check rule is performed before the keyless locking action succeeds. The first thing we do is try to use the key if the player is outside and has the key: this way, LOCK DOOR will turn automatically into LOCK DOOR WITH THE KEY, under circumstances where that is both possible and necessary.

The second thing is to check whether the player is outside but keyless, and, if so stop the action from being performed successfully. Here we print a failure message followed by the word "instead", which tells Inform that we've substituted some other outcome for the usual performance of the action.

Now we're reasonably sure that the player is only locking keylessly in the case that he is inside the Studio. (We might have to do a more thorough check for this if there were more than two rooms, but as it is, the player can only be in the Stairwell or in the Studio, so if we have ruled out the Stairwell, we are safe.) So now we want to add what happens when locking-without-a-key command succeeds:

```
Carry out locking keylessly:
    now the noun is locked.
```

That's it. We've just told Inform to make the door be locked. "Now..." syntax will be explained more thoroughly in the chapter on change. But we still haven't described to the player what just happened, so let's provide a description of that, too:

```
Report locking keylessly:
    say "You flip over the deadbolt to lock [the noun]."
```

And now we have to do a similar set of things for unlocking:

```
Understand "unlock [something]" as unlocking keylessly.
Unlocking keylessly is an action applying to one thing.


Check unlocking keylessly:
    if the noun is not a door, say "[The noun] is not
something you can lock." instead;
    if the noun is unlocked, say "[The noun] is already
unlocked." instead;
    if the player carries the brass key and the player is in
the Stairwell, try unlocking the noun with the brass key
instead;
    if the player is in the Stairwell, say "You can't unlock
the door from this side without the key." instead.


Carry out unlocking keylessly:
    now the noun is unlocked.


Report unlocking keylessly:
    say "You flip over the deadbolt to unlock [the noun]."


Test me with "unlock door / drop key / open door / out /
close door / lock door / open door / in / get key / out /
close door / lock door / unlock door".
```

Some (but not all) of this work is done for you if you like by the Locksmith extension. If you prefer, you can include that extension, then follow the documentation in order to implement the remainder of the scenario. Locksmith takes care of implementing the additional locking and unlocking actions, and provides some other conveniences.

ExampleOne Short Plank

A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

"One Short Plank"

The East Jungle is a room. The plank bridge is west of the East Jungle and east of the West Jungle. The plank is an open unopenable door. "A precarious plank bridge extends [if the location is West Jungle]east[otherwise]west[end if] across the chasm." The description of the plank is "Extremely fragile and precarious."

Instead of going through the plank when the player is carrying something:
    say "You step gingerly across the plank, which bows under your weight. But your meagre possessions are the straw which breaks the camel's back!";
    end the story.

After going through the plank:
    say "You step gingerly across the plank, grateful that you're not burdened.";
    continue the action.

There is a feather in the East Jungle.

But indeed, why stop there?

The gigantic stone ball is a thing in the West Jungle. It is pushable between rooms.

Before going through the plank with something:
    say "Surely you jest." instead.

Test me with "w / e / w / push ball e / e / get feather / w".

Example Elsie

A door that closes automatically one turn after the player opens it.

Suppose we want to create an automated door of the sort that closes when it isn't in use. A convenient way is to write a rule that fires "every turn when the sliding door was open". This will be true only if the door was open at the beginning of the turn: if the player just opened it this turn, it stays open, and if it was already closed, it stays closed. Thus:

```
"Elsie"

The axis-ward is a direction. The opposite of axis-ward
is hub-ward. Understand "aw" or "axisward" as axis-
ward.
The hub-ward is a direction. The opposite of hub-ward is
axis-ward. Understand "hw" or "hubward" as hub-ward.

The Ship's Bridge is a room.

The sliding door is a door. It is axis-ward from Bridge
and hub-ward from C Deck. The initial appearance is
"There is a door in this wall[if closed] -- or at least, the
potential for a door, since currently it is sealed,
distinguishable from the rest of the wall only by the
warning stripes on its surface[end if]."

Every turn when the sliding door was open:
    now the sliding door is closed;
    if the player can see the sliding door:
        say "The sliding door slips back into place, sealing
the wall as though there had never been a breach."
```

```
After opening the sliding door:
    say "You press the appropriate buttons, and a section
of wall slides away."



Test me with "open door / look / enter door / z".
```

 ExampleGaribaldi 1

Providing a security readout device by which the player can check on the status of all doors in the game.

Suppose we would like to allow the player to view the status of all the doors functioning in the game; and we want to identify those doors by mentioning which two rooms they connect. The following uses some techniques that will be covered in later chapters, but the basic idea may be obvious:

```
"Garibaldi"



The security readout is a device. The description of the
readout is "The screen is blank."



Instead of examining the switched on security readout:
    say "The screen reads: [fixed letter spacing]";
    say line break;
    repeat with item running through doors:
        say line break;
        say " [item] ([front side of the item]/[back side of the
item]): [if the item is
locked]LOCKED[otherwise]UNLOCKED[end if]";
    say variable letter spacing;
    say paragraph break.
```

It is more or less arbitrary which room winds up as the "front side" and which as the "back", but in this case it hardly matters.

The player carries the security readout.

The Docking Bay is a room. The inner airlock is a door. It is north of the Docking Bay and south of the Zocalo. The inner airlock is lockable and unlocked. The outer airlock is lockable and locked. It is a door. It is south of the Docking Bay and north of Space.

The quarantine seal is a door. It is west of the Zocalo and east of Medlab. Quarantine seal is locked.

The security pass unlocks the inner airlock. The player carries the security pass.

Test me with "x readout / turn on readout / x readout / lock inner airlock with security pass / x readout".

ExampleWhence?

A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

It would be fairly tedious reading to have a large game full of doors that describe themselves this way. Nonetheless, if we insisted we could use our knowledge of the map as leverage to make every door in the game describe itself automatically.

To do this, we make use of the phrase "direction of (the door) from (a room)" -- in this case, the direction of the door we're looking at when viewed from the player's location. Thus:

"Whence?"

The temporal vortex is an open door. It is west of Yesterday and east of Today.

The initial appearance of a door is usually "Nearby [an item described] leads [if the other side of the item described is visited][direction of the item described from the location] to [the other side][otherwise][direction of the item described from the location][end if]."

Test me with "w / e".

Yet a further variation on this, which can automatically understand "the east door" and "the west door" when appropriate, may be found in the example "Whither?".

Example: Vitrine

An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

"Vitrine"

Plaza View is a room. "Your uncle's apartment, on loan to you for viewing the parade and celebrations today. This would be more of a sacrifice on his part if he weren't currently yachting around Corfu."

The smart window is a device in Plaza View. It is fixed in place. "A vast smart window [if transparent]overlooks the park[otherwise]has turned to a sheet of hazy blue[end if]." The smart window can be transparent. The smart window is transparent. The description is "An

electrochromic device which changes shade and transparency in response to the application of current.

Curtains are so last year."

Note the "can be transparent" line. Devices ordinarily are not allowed to have transparency or opaqueness, but we can make an exception in this case. Without that line, attempts to change the transparency of the window will fail.

Carry out switching off the window: now the window is transparent.

Carry out switching on the window: now the window is opaque.

Instead of searching a transparent window: say "Isn't it lovely out there?"

Instead of searching an opaque window: say "The window is currently darkened."

Test me with "look through window / switch window / look through window / look".

ExampleEscape

Window that can be climbed through or looked through.

Suppose we want to offer the player a window he can climb through, instead of a boring ordinary door. Our window will be like a door in that it connects two rooms, appears in both places, and impedes movement

when it is shut. But we also want to add that we can look through it and see what lies on the other side; and we further want to understand "climb through window" or "jump through window" as attempts to pass through it.

We'll start by defining a couple of rooms and making the window a door between them.

```
"Escape"


Your Bedroom is a room. The bedroom window is a
door. It is west of Your Bedroom and east of the Grassy
Slope.
```

Now we have a "bedroom window" object which can be entered. Now, to catch the case where the player types "LOOK THROUGH WINDOW":

```
Instead of searching the window:
    say "Through the window, you make out [the other
side of the window]."
```

The other side of a door is always defined to be the room that we are not currently in when doing the check. When we are in the bedrooom, the other side will be the grassy slope, and vice versa. "Searching" is the action that occurs when the player attempts to LOOK THROUGH something. (To review what grammar gives rise to what actions, we can always consult the Actions portion of the Index.)

Next we want to cover the case where we climb through the window:

```
Instead of climbing the window:
    try entering the window.
```

And because "climb window" is understood but "climb THROUGH window" is not, we will have to borrow from the chapter on Understanding to add some new vocabulary to the game (and we'll add Jump too, while we're at it):

```
Understand "climb through [something]" as climbing.
```

```
Understand "jump through [something]" as climbing.
```

Now the final piece: Inform will already keep the player from going through a closed window, but it will say "You can't, since the bedroom window is in the way." This is probably not ideal, so we can replace the instruction thus:

```
Instead of going through the closed window:
    say "The window is shut: you'd break the glass."

Test me with "look through window / climb through
window / open window / climb through window / look
through window / close window / e / open window / e".
```

Example **Dinner is Served**

A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

```
"Dinner is Served"

Street in Kolonaki is a room. "There is a single round
table out on the street here, and a window more or less
at knee level looks down into the Olive Tree Gyro Shop,
which is partly basement."

The Street contains a round table. The table is scenery.
On the round table is a plate. On the plate are a gyro
and a mound of fresh potates. The plate is portable. The
potates and the gyro are edible. The description of
potates is "They'd be called french fries, at home, but
these are steak-cut and fried in olive oil." The
description of the gyro is "Dripping garlic-yogurt sauce."
```

```
Olive Tree Gyro Shop is inside from Street in Kolonaki.
Kostis is a man in the Gyro Shop. In the Shop is a
stand. On the stand is a rotating column of cooking lamb
flesh. In the shop is a closed, openable container called
a drinks refrigerator. The refrigerator contains a can of
Mythos beer and a can of Coke Light.
```

Here's the part that allows reaching through the window.

We replace the usual rule that says the player can never reach into a room with one that more specifically checks whether we are trying to reach through the window. If we aren't, we return the usual refusal. If we are, we return a custom refusal if the window is closed ("You can't reach through the closed window"), but allow access if the window is open.

```
The can't reach through closed window rule is listed
instead of the can't reach inside rooms rule in the
reaching inside rules.
```

```
This is the can't reach through closed window rule:
    let reaching through the window be false;
    if the container in question is a room and the
container in question is not the location:
        if the container in question is the Street and the
location is the Olive Tree Gyro Shop:
            now reaching through the window is true;
        if the container in question is the Gyro Shop and
the location is the Street:
            now reaching through the window is true;
        if reaching through the window is true:
            if the window is closed:
                say "You can't reach through the closed
window.";
                deny access;
            otherwise:
                allow access;
        otherwise:
            say "You can't reach into [the container in
question] from here.";
            deny access.
```

And the rest is window-dressing.

After looking when a room (called the next room) is adjacent:
    try examining the next room.

Instead of examining a supporter, say "On [the noun] [is-are a list of things on the noun]." Instead of examining an open container, say "In [the noun] [is-are a list of things in the noun]."

The window is a backdrop. It is in the Street and the Shop. The window can be openable. The window can be open. The window is openable and closed. Instead of searching the window in the Street: try examining the shop. Instead of searching the window in the Shop: try examining the street.

Understand "examine [any adjacent room]" as examining.

Instead of examining a room:
    say "Over in [the noun], you can see [a list of visible things in the noun]."

After deciding the scope of the player:
    if the player is in the Street, place the Shop in scope;
    if the player is in the Shop, place the Street in scope.

Test me with "examine shop / open refrigerator / open window / examine shop / open refrigerator / get beer / in / examine street / out / get gyro / close window / put gyro in refrigerator / open window / put gyro in refrigerator".

A cell window through which the player can see people who were in Port Royal in the current year of game-time.

Our protagonist is imprisoned in Port Royal, waiting out his years, and sometimes through the window of his cell he is able to see someone.

We are, however, obsessive about historical accuracy, so we provide a table of people who really lived in the city, together with the year in which their existence is attested. We want these people to appear in the description only in the year when they are known to have been present. (After all, mortality was high in Port Royal and new people were constantly arriving, so someone's presence one year is no guarantee of their continued existence the next.)

```
"Waiting for Godot, Chyrurgeon"

New New Prison is a room. "You have a not very
interesting room. Through the window you see passing
[current denizen]."

Instead of waiting:
    increment the current year;
    say "It is now the year [the current year].";
    try looking.

When play begins: now the right hand status line is
"[current year]".

Every turn:
    if the current year is 1692:
        say "It turns out you have remained imprisoned
until the great earthquake of 1692! Oops.";
        end the story.
```

```
Current year is a number that varies. The current year is 1664.

To say current denizen:
    repeat through the Table of Occupations and People:
        if the date attested entry is the current year:
            say "[nickname entry] [family entry], [trade entry]";
            blank out the whole row;
            rule succeeds;
    say "absolutely no one".
```

It is possible to look up a row corresponding to, say, a specific year value using "listed in", but repeat through is convenient here because we know that we will never wind up trying to print entries when no row can be successfully selected.

```
Table of Occupations and People
```

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

Test me with "wait / wait / wait / wait / wait / wait / wait / wait / wait / wait".

---

181 Example **A Haughty Spirit** *WI*

Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

Suppose we have a game in which the player can climb through windows which overlook rooms below. We want him to be allowed to climb out

windows to reach a room on the same level or at most one level lower than the one he's on; otherwise, he should get a refusal, saying that he would break his neck.

To figure out the height distance between the start room and the destination room, we might have a repeat loop look at all the directions one has to follow along the "best route" path between the two rooms, and record any ups and downs; then subtract the number of "up" steps from the number of "down" steps, and report what remains.

```
"A Haughty Spirit"


To decide what number is the distance (first place - a
room) rises above (second place - a room):
    let the total distance be the number of moves from
the first place to the second place;
    if the total distance is less than 1, decide on 0;
    let count of down moves be 0;
    let count of up moves be 0;
    let next place be the first place;
    repeat with counter running from 1 to the total
distance:
        let the way be the best route from the next place to
the second place;
        if the way is down, let count of down moves be the
count of down moves plus 1;
        if the way is up, let the count of up moves be the
count of up moves plus 1;
        let next place be the room the way from next place;
    let the decision be the count of down moves minus
the count of up moves;
    decide on the decision.
```

Now we just have to create windows and some action rules for interacting with them...

```
A window is a kind of thing. A window is always fixed in
place. A window can be open or closed. A window is
usually closed. A window can be openable or
unopenable. A window is usually openable.
```

```
Understand "climb through [something]" as entering.
Understand "jump through/out [something]" as entering.
```

```
Before entering a closed window:
    say "[The noun] would have to be opened first."
instead.
```

```
Instead of entering a window:
    if the noun overlooks a room (called the far side):
        let fall be the distance the location rises above the
far side;
        if fall is greater than 1, say "You'd break your
neck." instead;
        say "You tumble into [the far side].";
        move the player to the far side;
    otherwise:
        say "There's nowhere to go."
```

```
Instead of examining a window:
    say "[The noun] [if the noun is open]opens
over[otherwise]gives a view of[end if] [the list of rooms
overlooked by the noun]."
```

Here we must anticipate a little from the chapter on Relations, and provide ourselves with a way of keeping track of how windows and rooms relate to one another:

```
Overlooking relates various windows to various rooms.
The verb to overlook means the overlooking relation.
The initial appearance of a window is usually "[The item
described] overlooks [the list of rooms overlooked by the
item described]."
```

```
The Square Keep is above the Winding Staircase. The
Winding Staircase is above the Motte. A crown and a
broken sword are in the Motte. The Bailey is west of the
Motte.
```

```
The long window is in the Keep. The long window
overlooks the Bailey and the Motte. The narrow window
is in the Winding Staircase. The narrow window
overlooks the Bailey.
```

```
Test me with "jump through window / open window /
jump through window / d / x narrow window / open
window / climb through window / e / up / down".
```

We could then add rules to allow the player to look through windows and see things in the rooms below, but that would require more material from later chapters.

---

194 Example The Dark Ages Revisited *WI*

An electric light kind of device which becomes lit when switched on and dark when switched off.

This will be explored more in subsequent examples, but one of the things we can do with carry out rules is extend the function of existing commands so that they do more, or have special effects in specific situations. For instance, suppose we want to have a class of electric light:

```
"The Dark Ages Revisited"
```

```
An electric light is a kind of device. Carry out switching
on an electric light: now the noun is lit. Carry out
switching off an electric light: now the noun is unlit.
```

This will not affect the behavior of any other devices when switched; it will also not change the way in which switching lights on and off is reported. The player will still see "You switch the sodium lamp on." or the like. In this case that is probably what we want. If we wanted a special way of describing turning on electric lights as opposed to all other devices, we could also add an after rule for the electric light class. Adding this rule to the carry out train does guarantee, though, that in no case will we manage to make the lamp lit without actually making it switched on

(or vice versa).

```
The Stooped Corridor is a room. "A low, square-cut
corridor, running north to south, stooping you over."
```

```
The sodium lamp is an electric light in the Stooped
Corridor. "[if switched on]The sodium lamp squats on
the ground, burning away.[otherwise]The sodium lamp
squats heavily on the ground.[end if]". The description is
"It is a heavy-duty archaeologist's lamp, [if switched
off]currently off.[otherwise]blazing with brilliant yellow
light.[end if]"
```

```
Instead of burning the sodium lamp, try switching on the
lamp.
```

So far so easy. Since we've built the description of its light or darkness into the lamp's description, though, we may want to get rid of the "...is switched on" line that automatically follows when we look at something. For this we do need to borrow from a later chapter:

```
The examine devices rule is not listed in the carry out
examining rules.
```

```
Test me with "x lamp / switch lamp on / look / x lamp".
```

```
"Hymenaeus"
```

A torch is kind of thing. Understand the lit property as describing a torch. Understand "lighted" or "flaming" or "burning" as lit. Understand "extinguished" as unlit. A torch is usually lit.

Before printing the name of a lit torch, say "flaming ".
Before printing the name of an unlit torch, say "extinguished ".

The Wedding Procession is a room.

Orpheus is a man in the Wedding Procession. Orpheus carries a torch.
Eurydice is a woman in the Wedding Procession. Eurydice carries a torch.

Rule for writing a paragraph about someone (called target):
    say "[The target] carries [a list of things carried by the target]."

Every turn:
    if a random chance of 1 in 2 succeeds and a torch is lit:
        let target torch be a random lit torch;
        now the target torch is unlit;
        say "Aquilo blows down from the north, extinguishing the torch carried by [the holder of the target torch]."

Instead of examining a lit torch:
    say "It casts a bright glow over [the holder of the noun]."
Instead of examining an unlit torch:

say "[The holder of the noun] is looking at it disconsolately, obviously worried about the omens."

Test me with "z / z / z / look / x flaming torch / x extinguished torch".

---

352 **Example** **Reflections** *WI*

Emphasizing the reflective quality of shiny objects whenever they are described in the presence of the torch.

"Reflections"

Behind the Waterfall is a room. "Though one wall of the cave is open to the waterfall, the quantity of water is so great that barely any light comes through from the outside." Behind the Waterfall is dark.

Surface is a kind of value. The surfaces are shiny and dull. A thing has a surface. A thing is usually dull.

The player carries a reflecting ball, a canopic jar, an abacus, a plumbline, a piece of chalk, and a torch. The reflecting ball is shiny.

Aladdin's lamp is a shiny thing in Behind the Waterfall.

Brightness is a kind of value. The brightnesses are guttering, weak, radiant and blazing. The torch has a brightness. The torch is blazing. The torch is lit.

```
Understand "blow out [something]" or "blow
[something]" or "extinguish [something]" as blowing out.
Blowing out is an action applying to one thing.

Carry out blowing out: say "Futile."

Instead of blowing out the torch:
    now brightness of torch is the brightness before the
brightness of the torch;
    say "The light of the torch dies to [brightness of
torch]."

Instead of blowing out the guttering torch:
    say "Fool! Do you want to put it out entirely?"

Rule for writing a paragraph about a shiny thing:
    say "The [brightness of the torch] light of [the torch]
reflects in the surface[if the number of shiny things in
the location > 1]s[end if] of [the list of shiny things in the
location]."

Before printing the name of the torch while writing a
paragraph about something:
    if the torch is in the location, say "fallen ".

Test me with "drop ball / look / blow torch / look / drop
torch / look".
```

ExamplePeeled

Two different approaches to adjusting what the player can interact with, compared.

Suppose we have a situation where the player is in darkness, but is allowed to feel and interact with (except for examining) any large objects. In that case, we write a scope rule that puts those large objects into scope all the time, and trust the "requires light" aspect of verbs like examining to prevent the player from doing any actions that he shouldn't:

```
"Peeled"


A thing can be large or small.


Before touching a large thing when in darkness: say
"You grope for [the noun]..."


After deciding the scope of the player:
    repeat with item running through large things in the
location:
        place item in scope.


Some generic surroundings are backdrop. They are
everywhere. Understand "walls" or "wall" or "ceiling" or
"ground" or "floor" or "area" or "room" or "here" as the
generic surroundings. Instead of touching the generic
surroundings: say "You encounter nothing
extraordinary." Instead of touching the generic
surroundings when in darkness: say "You try feeling
your way around and reach [a list of large things in the
location]." After deciding the scope of the player when in
darkness: place the surroundings in scope.


The Room of Mystery is a dark room. The bearskin rug
is a large thing in the Room of Mystery. Instead of
touching the rug: say "It feels furry!"
```

```
The peeled grape is a small thing in the Room of
Mystery. Instead of touching the peeled grape: say
"Gosh, is that an eyeball?"



Test me with "feel floor / feel rug / eat rug / examine rug
/ get grape".
```

Sadly, because the grape is small, the player will never encounter this horror.

Alternatively, suppose we have a situation in which the player can use one command to interact with a kind of thing that isn't normally in scope. It's usually most convenient to write the "understand" rule appropriately rather than use the scope activity.

(Note that we define "inquiring about" as applying to one *visible* thing; otherwise we would be required to be able to touch the catsuit in order to inquire about it. More on this restriction may be found in the Advanced Actions chapter on the topic of visible, touchable, and carried things.)

```
"Peeled"



Mr Steed's Flat is a room.



Understand "ask about [any subject]" as inquiring about.
A subject is a kind of thing. The skintight catsuit is a
subject. Inquiring about is an action applying to one
visible thing.



Carry out inquiring about something:
    say "'What can you tell me about [the noun]?' you
demand. Mr Steed raises his eyebrows, but does not
reply."
```

```
Test me with "ask about catsuit / x catsuit".
```

All this said, there do arise certain complex situations when we want an activity-specific scoping.

Example **Down Below** *WI*

A light switch which makes the room it is in dark or light.

Suppose we want to have a room with a light switch. Turning the switch off makes the room go dark; turning it on restores the light. This kind of switch is an obvious candidate as a device.

```
"Down Below"

Terrifying Basement is a room. The light switch is a
switched on device in the Terrifying Basement. It is fixed
in place.
```

Here we define our light switch, and we also make it start out as "switched on". The Terrifying Basement will also start out lit (as all rooms do, by default, unless we specifically say that they are dark). We further say that it is fixed in place to avoid the ludicrous possibility of the player picking it up and carrying it away.

Next we add some instructions to control how turning the light switch on and off affects the room light. These borrow from later chapters on actions, but the gist may be obvious anyway:

```
Carry out switching off the light switch: now the
Terrifying Basement is dark.

Carry out switching on the light switch: now the
Terrifying Basement is lighted.
```

Inform already has the idea of light and darkness built in; we will see

more about this later, and the Phrasebook (in the Index tab) also contains a list of all the adjectives (lighted, dark, etc) which are important to use here.

Speaking of the Index, the Actions tab contains a list of all the grammar that can be used to activate a given command: for instance, the switching action responds to "switch [something]" or "turn on [something]". In this case, we may want to give the player an extra option or two. It would be pretty natural for a player to try >FLIP SWITCH, so let's add that in:

```
Understand "flip [something switched off]" as switching
on. Understand "flip [something switched on]" as
switching off. Understand "flip [something]" as switching
on.
```

The nuances of this will be explored in the chapter on Understanding. What is useful to know here is that we have taught Inform to understand that >FLIP LIGHT SWITCH means to turn it on when the switch is already off; if the switch is already on, FLIP SWITCH means to turn the switch off. Depending on the kind of device we are modeling (button? lever? dial?), we might want to write similar lines for commands such as PUSH, PRESS, PULL, TURN, and so on.

Finally, we need to deal with a special case. In general, the player cannot interact with other things in a dark room because he can't see them, but if we adhered strictly to this it would be impossible for him to find the light switch to turn it back on. So we need something from the chapter on Activities to change this:

```
After deciding the scope of the player when the location
is the Terrifying Basement:
    place the light switch in scope.


Upstairs is above the Terrifying Basement.


Test me with "turn off light / look / flip light switch".
```

Flickering lantern-light effects added to the Undertomb.

"Undertomb 2"

Section 1 - Procedure

Brightness is a kind of value. The brightnesses are guttering, weak, radiant and blazing.

A brightness can be adequate or inadequate. A brightness is usually adequate. Guttering is inadequate.

Temperature is a kind of value. 100C specifies a temperature.

A brightness has a temperature. The temperature of a brightness is usually 700C. The temperature of blazing is 1400C. The temperature of radiant is 1100C.

A dead end is a kind of room with printed name "Dead End" and description "This is a dead end, where crags in the uneven rock are caught by the [brightness of the lantern] flame you hold aloft. Despite [river sound] there is no sign of the stream." A dead end is usually dark.

A dead end has some text called river sound. The river sound of a dead end is usually "a faint whispering of running water".

Section 2 - Scenario

The Undertomb is a dark room. "From this dim cross-groined room, exits depart east, south, and northwest."

East is a dead end. South is a dead end with printed name "Collapsed Dead End".

Northwest is a dead end called the Tortuous Alcove. The Tortuous Alcove has river sound "a gurgle of running water".

The player carries a book. The description of the book is "[if the brightness of the lantern is adequate]Many secrets are now yours.[otherwise]No, the print's too tiny by this awful light."

In the Undertomb is a lantern. It is lit. The lantern has a brightness. The lantern is blazing. The description of the lantern is "The lantern shines with a flame at [temperature of the brightness of the lantern]."

After waiting in the Tortuous Alcove when the brightness of the lantern is not guttering:
    now the lantern is the brightness before the brightness of the lantern;
    say "You wait so long that your lantern dims a bit."

Test me with "east / west / get lantern / east / west / south / north / northwest / read book / x lantern / z / x lantern / read book / look / z / x lantern / read book / look / z / x lantern / read book / look".

ExampleHohmann Transfer

Changing the way dark rooms are described to avoid the standard Inform phrasing.

Inform automatically keeps track of light and darkness, handling such questions as whether a room is lit, whether the player can see any light sources, etc., and then managing the descriptions accordingly. When the room is dark and no light sources are visible, the player is said to be "in darkness".

If we don't specify otherwise, Inform will describe our surroundings in a dark room thus:

```
Darkness
It is pitch dark, and you can't see a thing.
```

This is fine in many situations, but we may sometimes want to replace this phrase with something else.

```
"Hohmann Transfer"

The Western Hemisphere is a dark room. "The cloud
mass covers much of the land on this side of the planet,
and a particularly nasty storm is brewing off to the
south."

The Eastern Hemisphere is west of the Western
Hemisphere. The Eastern Hemisphere is east of the
Western Hemisphere. The Eastern Hemisphere is north
of the Western Hemisphere. The Eastern Hemisphere is
south of the Western Hemisphere. "This side of the
planet is more ocean than land, with only two continents
worthy of the name, and a volcanic archipelago in the
north seas."
```

```
Use full-length room descriptions.

Rule for printing the description of a dark room:
    say "It's night on this side of the planet, so you can
make out only the glow of urbanized areas along the
seacoasts." instead.

Rule for printing the name of a dark room:
    say "Dark Side" instead.
```

And now a few minor refinements so that we can see what happens when one room becomes dark and the other light:

```
Carry out going:
    say "You fire the thrusters and loop around to the
other side of the planet before settling into a new
geosynchronous orbit. Six months and one minute
later..."

The time of day is 4:55 PM.

At 5 PM:
    now the Eastern Hemisphere is dark;
    now the Western Hemisphere is lit.

Rule for printing the announcement of darkness:
    say "The planet abruptly spins itself over, exposing its
cool underbelly to the sun."

Test me with "e / z / z / w / z / z / e".
```

Suppose we're simulating a situation where the player needs to travel through lit areas only, but we want to give him some hints about which way might be safe. Here we'll find our best route through light-filled rooms.

The slightly tricky part is that it's not necessarily easy to tell whether a room has a lamp in it. We may say "if the Crypt is lighted", but that only tells us whether it has been declared to be inherently lighted or dark, not whether it happens to contain a light source that the player would be able to see if he went in.

The easiest way to get around this is to create an object -- the light-meter; place it in the target location; and check whether it "can see" a lit object. This preserves all the usual rules about open and closed containers, transparency, etc.

```
"Unblinking"


Section 1 - Procedure


The light-meter is a privately-named scenery thing.


Definition: a room (called the target room) is light-filled:
    if the target room is lighted:
        yes;
    move the light-meter to the target room;
    let the answer be false;
    if the light-meter can see a lit thing:
        now the answer is true;
    now the light-meter is nowhere;
    decide on the answer.
```

That done, we're free to use our best-route phrases to choose a particular route.

Section 2 - Scenario

The Tomb of Angels is a room. "This ancient underground tomb is lightless but for a few shafts from the surface. Everywhere in the shadows are carved angels, their faces worn away by water and pollution, their wings little more than nubs."

The Upward Path is above the Tomb of Angels. It is dark. "The staircase switches back on itself many times as it ascends towards the Crash Site."

A container called the sarcophagus is in the Upward Path. It is closed and openable. "A sarcophagus rests in the niche here, [if open]the lid pushed aside[otherwise]the lid firmly in place[end if]."

The Crash Site is above the Upward Path. "The ceiling has wholly caved in here, and the belly of the spaceship above you is visible -- including the escape hatch."

A candle is a kind of thing. A candle is usually lit. The player carries four candles.

After looking when the location is not the Crash Site:
    if the best route from the location to the Crash Site through light-filled rooms is a direction (called next way):
        say "It looks like there's a safe, lit path [if the number of moves from the location to the Crash Site through light-filled rooms is 1]straight[otherwise]if you go[end if] [next way].";
    otherwise:
        say "It looks like there is no fully lit path from here to the Crash Site."

Test me with "up / drop candle / down / up / take the candle / open sarcophagus / put candle in sarcophagus / down / up / close sarcophagus / d".

An important word of caution: this method would give false negatives if there were a backdrop lightsource, such as the moon, providing light to the Upward Path. This is because backdrops are actually moved around the map by Inform during play, following the player around. So if the moon backdrop is in the Crash Site with the player, it will not be in the Upward Path as well -- even if it's scheduled to move there as soon as the player does.

ExampleZorn of Zorna

Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

```
"Zorn of Zorna"

Visibility rule:
    if examining:
        if the detail of the noun is fine and the number of
visible lit candles is less than 5, there is insufficient light;
        if the detail of the noun is ordinary and the number
of visible lit candles is less than 3, there is insufficient
light;
    there is sufficient light.

Detail is a kind of value. The details are fine, ordinary,
and gross. A thing has detail.

A candle is a kind of thing. Before printing the name of a
candle while not burning or blowing out: say "[if lit]lit
[otherwise]unlit [end if]". A candle is usually lit. Before
printing the plural name of a candle while not burning or
```

blowing out: say "[if lit]lit [otherwise]unlit [end if]". A candle is usually lit. Understand the lit property as describing a candle. A candle is usually gross.

Instead of burning a candle: now the noun is lit; say "You light [the noun]."

Understand "blow out [something]" or "extinguish [something]" or "put out [something]" as blowing out. Understand the command "snuff" as "extinguish". Blowing out is an action applying to one thing.

Understand "burn [unlit candle]" as burning.

Instead of blowing out a candle:
    now the noun is unlit;
    say "You put out [the noun]."

Rule for printing a refusal to act in the dark:
    if we are examining something, say "The details of [the noun] are too fine to make out in the light of only [the number of visible lit candles in words] candle[s]." instead.

Every turn when the Todal is visible:
    if the number of visible lit candles is greater than 1:
        say "The brightness of the room wakens the Todal from slumber, and with you unarmed...";
        end the story;
    otherwise:
        say "Todal sleeps fitfully, troubled by even that faint light."

A room is usually dark.

The Palace is a room. "The Duke is out; the way is clear. East is Saralinda's Chamber; north, a hallway zigs and zags down to the gate that leads out." A finely-written placard is in the Palace. "A finely-written placard is on the wall next to this exit." The placard is fine. The description of the placard is "You read: 'Beware the Todal: its bite is worse than its gleep.

No more than one candle!'"

The candle-stand is a supporter in the Palace. Understand "stand" as the candle-stand. The description of the candle-stand is "The candle-stand is a tall metal branch for holding lights, but someone has quite practically added casters to the bottom." It is pushable between rooms. Three candles are on the candle-stand. Instead of removing something from the candle-stand: say "[The noun] is fixed quite firmly in place." Instead of taking something which is on the candle-stand: say "[The noun] won't come out of the holder." Instead of putting something on the candle-stand: say "[The candle-stand] is full."

Saralinda's Chamber is east of the Palace. "Now that Saralinda herself is gone, there is no real radiance in this place." Two unlit candles are in Saralinda's Chamber.

A large-print romantic novel is in Saralinda's Chamber. The novel is ordinary. The description of the novel is "'She Was Only The Chimney-Sweep's Daughter', by Marie Swelldon."

The Zig-Zag Hallway is north of the Palace. "The Hallway goes left, then right, then left again..." Two unlit candles are in the Hallway.

Todal is an animal in the Zig-Zag Hallway.

Rule for printing the description of a dark room when the Todal is in the location:
    try listening.

Instead of listening when in darkness and the Todal is in the location:
    say "In the darkness something softly gleeps."

Instead of going north from the Hallway when in darkness:
    say "You stumble and cannot find your way."

North of the Hallway is Freedom. Instead of going to Freedom: say "You make it out into the cool night air at last!"; end the story finally.

Test me with "examine placard / get placard / n / listen / n / s / examine candle-stand / push candle-stand east / examine novel / get unlit candle / light it / light unlit candle / examine placard / push candle-stand west / e / examine novel / w / n / n".

ExampleFour Stars 1

An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead.

"Four Stars"

A thing has some text called sound. The sound of a thing is usually "silence".

The report listening rule is not listed in the report listening to rules.

Carry out listening to something:
    say "From [the noun] you hear [the sound of the noun]."

Instead of listening to a room:
    if an audible thing can be touched by the player, say "You hear [the list of audible things which can be touched by the player].";
    otherwise say "A merciful peace prevails."

Definition: a thing is audible if the sound of it is not "silence".

Before printing the name of something audible while listening to a room:
    say "[sound] from the "

A thing has some text called scent. The scent of a thing is usually "nothing".

The report smelling rule is not listed in the report smelling rulebook.

Carry out smelling something:
    say "From [the noun] you smell [scent of the noun]."

Instead of smelling a room:
    if a scented thing can be touched by the player, say "You smell [the list of scented things which can be touched by the player].";
    otherwise say "The place is blissfully odorless."

Definition: a thing is scented if the scent of it is not "nothing".

Before printing the name of something scented while smelling a room: say "[scent] from the "

The Waning Moon Resort is a dark room. "A spacious room with a flagstone floor, and a dreamcatcher hung over the king-size bed." The dreamcatcher is scenery in the Resort. The description is "The usual web of threads and crystals, feathers and beads." Instead of taking the dreamcatcher, say "Ah, ah -- you might be tempted to take it as a souvenir, except that the price list in the minibar clearly states they charge $65 apiece if you walk off with one. Cheaper than stealing the Frette bathrobes, but still probably not a good idea."

The king-size bed is an enterable supporter in the Resort. The description is "200-thread-count Egyptian cotton sheets, according to the website. You would make fun, only they really are extraordinarily comfortable." The player is on the bed. A Lindt chocolate is on the bed. It is edible. The scent of the chocolate is "chocolate-hazelnut smell".

Instead of exiting: say "You are too weary to move."

The suitcase is an openable closed container in the Resort.

An electric light is a kind of device. Carry out switching on an electric light: now the noun is lit. Carry out switching off an electric light: now the noun is unlit. Understand "light" as an electric light.

The solar lamp is an electric light in Waning Moon Resort. The description is "Specially designed to give light in a spectrum resembling sunlight, to improve the mood and make a person energetic." The lamp is switched on and lit.

An electric noisemaker is a kind of device. An electric noisemaker has some text called usual sound. The usual sound of an electric noisemaker is usually "beepbeepbeep". Carry out switching on an electric noisemaker: now the sound of the noun is the usual sound of the noun. Report switching on an electric noisemaker: say "[The noun] goes [usual sound of the noun]!" instead. Report switching off an electric noisemaker: say "You switch off [the noun], silencing the [usual sound of the noun]." instead.

Carry out switching off an electric noisemaker: now the sound of the noun is "silence".

The bedside table is in the Resort. The table supports a potted plant and a Bose speaker. The scent of the potted plant is "rosemary"

The Bose speaker is an electric noisemaker. The usual sound of the speaker is "soothing whalesong". The

sound of the speaker is "soothing whalesong". The speaker is switched on.

Instead of touching a device: say "You feel the surface of [the noun] and discover the switch."

Instead of touching a scented thing: say "The brush of your fingers stirs loose a fresh cloud of [scent of the noun] smell."

Rule for printing the description of a dark room: try listening; try smelling; rule succeeds.

Instead of examining an audible thing while in darkness: try listening to the noun. Instead of examining something while in darkness: try touching the noun.

Before touching something when in darkness:
    say "You grope about..."

After inserting the plant into something:
    say "You unceremoniously dump [the noun] into [the second noun], hoping it sustains no important damage thereby."

Before printing the name of a dark room: if the player can touch an audible thing, say "Noisy "; if the player can touch a scented thing, say "Perfumed ".

Visibility rule when in darkness:
    if examining something, there is sufficient light;
    there is insufficient light.

Rule for printing the announcement of darkness: say "It is now pleasantly lightless in here." instead.

Rule for deciding the scope of the player while in darkness: place the location in scope.

To decide whether in daylight:
    if in darkness, no;
    yes.

Instead of sleeping when in daylight:
    say "You've never been able to sleep with the light on."

Instead of sleeping when the player can touch an audible thing (called the irritant):
    say "The steady [sound of the irritant] from [the irritant] prevents your slumber."

Instead of sleeping when the player can touch a scented thing (called the irritant):
    if the irritant is chocolate, say "The smell of chocolate continues to tantalize you, keeping you from sleep.";
    otherwise say "You sniffle. [The irritant] is probably acting on your allergies."

Instead of sleeping:
    say "You slip easily into the arms of Morpheus.";
    end the story finally saying "At last..."

When play begins:
    say "You have at last escaped from the airport and

gotten through customs; survived an unnerving taxi ride over icy highways; stared down the impertinent concierge; endured the bellhop's catalog of features in your room; and achieved, finally, a moment of peace. Time for a good night's slumber!"

Test me with "x dreamcatcher / switch lamp off / look / sleep / eat chocolate / sleep / get plant / examine plant / open suitcase / put plant in suitcase / close suitcase / sleep / look / examine bose / switch bose off / sleep".

ExampleCloak of Darkness

Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

"Cloak of Darkness" is a brief example game that has been implemented in nearly every IF system currently used. It hasn't got much claim to complexity or richness, but it does exemplify many of the standard things one might want an IF language to be able to do: define descriptions and the results of actions, assign synonyms to nouns, create new verbs, handle darkness, track repeated acts, and so on.

Here is what the game looks like in Inform:

"Cloak of Darkness"

The story headline is "A basic IF demonstration."

Use scoring.

The maximum score is 2.

Whatever room we define first becomes the starting room of the game, in

the absence of other instructions:

```
Foyer of the Opera House is a room. "You are standing
in a spacious hall, splendidly decorated in red and gold,
with glittering chandeliers overhead. The entrance from
the street is to the north, and there are doorways south
and west."
```

```
Instead of going north in the Foyer, say "You've only just
arrived, and besides, the weather outside seems to be
getting worse."
```

We can add more rooms by specifying their relation to the first room. Unless we say otherwise, the connection will automatically be bidirectional, so "The Cloakroom is west of the Foyer" will also mean "The Foyer is east of the Cloakroom":

```
The Cloakroom is west of the Foyer. "The walls of this
small room were clearly once lined with hooks, though
now only one remains. The exit is a door to the east."
```

```
In the Cloakroom is a supporter called the small brass
hook. The hook is scenery. Understand "peg" as the
hook.
```

Inform will automatically understand any words in the object definition ("small", "brass", and "hook", in this case), but we can add extra synonyms with this sort of Understand command.

```
The description of the hook is "It's just a small brass
hook, [if something is on the hook]with [a list of things
on the hook] hanging on it[otherwise]screwed to the
wall[end if]."
```

This description is general enough that, if we were to add other hangable items to the game, they would automatically be described correctly as well.

```
The Bar is south of the Foyer. The printed name of the
bar is "Foyer Bar". The Bar is dark. "The bar, much
rougher than you'd have guessed after the opulence of
the foyer to the north, is completely empty. There seems
to be some sort of message scrawled in the sawdust on
the floor."


The scrawled message is scenery in the Bar.
Understand "floor" or "sawdust" as the message.


Neatness is a kind of value. The neatnesses are neat,
scuffed, and trampled. The message has a neatness.
The message is neat.
```

We could if we wished use a number to indicate how many times the player has stepped on the message, but Inform also makes it easy to add descriptive properties of this sort, so that the code remains readable even when the reader does not know what "the number of the message" might mean.

```
Instead of examining the message:
    increment score;
    say "The message, neatly marked in the sawdust,
reads...";
    end the story finally.
```

This second rule takes precedence over the first one whenever the message is trampled. Inform automatically applies whichever rule is most specific:

```
Instead of examining the trampled message:
    say "The message has been carelessly trampled,
making it difficult to read. You can just distinguish the
words...";
    end the story saying "You have lost".
```

This command advances the state of the message from neat to scuffed and from scuffed to trampled. We can define any kinds of value we like and advance or decrease them in this way:

```
Instead of doing something other than going in the bar
when in darkness:
    if the message is not trampled, now the neatness of
the message is the neatness after the neatness of the
message;
    say "In the dark? You could easily disturb something."


Instead of going nowhere from the bar when in
darkness:
    now the message is trampled;
    say "Blundering around in the dark isn't a good idea!"
```

This defines an object which is worn at the start of play. Because we have said the player is wearing the item, Inform infers that it is clothing and can be taken off and put on again at will.

```
The player wears a velvet cloak. The cloak can be hung
or unhung. Understand "dark" or "black" or "satin" as the
cloak. The description of the cloak is "A handsome
cloak, of velvet trimmed with satin, and slightly
splattered with raindrops. Its blackness is so deep that it
almost seems to suck light from the room."


Carry out taking the cloak:
    now the bar is dark.


Carry out putting the unhung cloak on something in the
cloakroom:
    now the cloak is hung;
    increment score.


Carry out putting the cloak on something in the
cloakroom:
    now the bar is lit.
```

```
Carry out dropping the cloak in the cloakroom:
    now the bar is lit.


Instead of dropping or putting the cloak on when the
player is not in the cloakroom:
    say "This isn't the best place to leave a smart cloak
lying around."


When play begins:
    say "[paragraph break]Hurrying through the rainswept
November night, you're glad to see the bright lights of
the Opera House. It's surprising that there aren't more
people about but, hey, what do you expect in a cheap
demo game...?"


Understand "hang [something preferably held] on
[something]" as putting it on.


Test me with "s / n / w / inventory / hang cloak on hook /
e / s / read message".
```

And that's all. As always, type TEST ME to watch the scenario play itself out.

Example **The Undertomb 1**

WI

A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

This example creates a kind of room, "dead end", and gives each a textual description called its "river sound". Note the use of a text substitution to fill in the appropriate river sound text in each dead end.

```
"The Undertomb"
```

A dead end is a kind of room with printed name "Dead End" and description "This is a dead end. You'll have to go back the way you came, consoled only by [river sound]." A dead end is usually dark.

The Undertomb is a dark room. East is a dead end. South is a dead end with printed name "Collapsed Dead End". Northwest is a dead end called the Tortuous Alcove. In the Undertomb is the lantern. It is lit.

A dead end has some text called river sound. The river sound of a dead end is usually "a faint whispering of running water". The Tortuous Alcove has river sound "a gurgle of running water".

Test me with "get lantern / e / w / s / n / nw / se".

Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

As we have seen, a well-written understand rule will often solve the problem of allowing the player to apply specific actions to objects not normally in scope. When we need to adjust scope for some other reason than reading the player's command, though, "deciding the scope of..." may come in handy.

For instance, suppose we wanted to extend Four Stars 1 to add a tomcat on the balcony that will be heard whenever the player listens from the next room, as in:

>listen
You hear the soothing whalesong from the Bose speaker and the yowling from the tomcat.

To do this, we need to make sure that in the rule that assembles our listening description,

```
Instead of listening to a room:
    if an audible thing can be touched by the player, say
"You hear [the list of audible things which can be
touched by the player].";
    otherwise say "A merciful peace prevails."
```

now includes the tomcat in the "list of audible things which can be touched by the player".

To this end, we're going to change the way we assess scope, but only during the listening action. Otherwise the tomcat remains in the other room and off-limits. The new source text is marked out below:

```
"Four Stars"


Section 1 - Procedure


A thing has some text called sound. The sound of a
thing is usually "silence".


The report listening rule is not listed in the report
listening to rules.


Carry out listening to something:
    say "From [the noun] you hear [the sound of the
noun]."


Instead of listening to a room:
    if an audible thing can be touched by the player, say
"You hear [the list of audible things which can be
touched by the player].";
```

```
    otherwise say "A merciful peace prevails."

Definition: a thing is audible if the sound of it is not
"silence".

Before printing the name of something audible while
listening to a room:
    say "[sound] from the "

A thing has some text called scent. The scent of a thing
is usually "nothing".

The report smelling rule is not listed in the report
smelling rulebook.

Carry out smelling something:
    say "From [the noun] you smell [scent of the noun]."

Instead of smelling a room:
    if a scented thing can be touched by the player, say
"You smell [the list of scented things which can be
touched by the player].";
    otherwise say "The place is blissfully odorless."

Definition: a thing is scented if the scent of it is not
"nothing".

Before printing the name of something scented while
smelling a room: say "[scent] from the "
```

Here is our addition:

After deciding the scope of the player while listening or sleeping or looking:
    if in darkness:
        repeat with locale running through adjacent rooms:
            place locale in scope.

A reaching inside rule while listening or sleeping or looking:
    rule succeeds.

Section 2 - Scenario

The Waning Moon Resort is a dark room. "A spacious room with a flagstone floor, and a dreamcatcher hung over the king-size bed." The dreamcatcher is scenery in the Resort. The description is "The usual web of threads and crystals, feathers and beads." Instead of taking the dreamcatcher, say "Ah, ah -- you might be tempted to take it as a souvenir, except that the price list in the minibar clearly states they charge $65 apiece if you walk off with one. Cheaper than stealing the Frette bathrobes, but still probably not a good idea."

And now our threat to the player's peace:

The Balcony is outside from the Resort. In the Balcony is a tomcat. The sound of the tomcat is "yowling". After printing the name of the tomcat when the tomcat is not visible: say " outside on the balcony".

From here we continue with the same scenario as before:

The king-size bed is an enterable supporter in the Resort. The description is "200-thread-count Egyptian cotton sheets, according to the website. You would make fun, only they really are extraordinarily comfortable." The player is on the bed. A Lindt

chocolate is on the bed. It is edible. The scent of the chocolate is "chocolate-hazelnut smell".

Instead of exiting: say "You are too weary to move."

The suitcase is an openable closed container in the Resort.

An electric light is a kind of device. Carry out switching on an electric light: now the noun is lit. Carry out switching off an electric light: now the noun is unlit. Understand "light" as an electric light.

The solar lamp is an electric light in Waning Moon Resort. The description is "Specially designed to give light in a spectrum resembling sunlight, to improve the mood and make a person energetic." The lamp is switched on and lit.

An electric noisemaker is a kind of device. An electric noisemaker has some text called usual sound. The usual sound of an electric noisemaker is usually "beepbeepbeep". Carry out switching on an electric noisemaker: now the sound of the noun is the usual sound of the noun. Report switching on an electric noisemaker: say "[The noun] goes [usual sound of the noun]!" instead. Report switching off an electric noisemaker: say "You switch off [the noun], silencing the [usual sound of the noun]." instead.

Carry out switching off an electric noisemaker: now the sound of the noun is "silence".

The bedside table is in the Resort. The table supports a

potted plant and a Bose speaker. The scent of the potted plant is "rosemary"

The Bose speaker is an electric noisemaker. The usual sound of the speaker is "soothing whalesong". The sound of the speaker is "soothing whalesong". The speaker is switched on.

Instead of touching a device: say "You feel the surface of [the noun] and discover the switch."

Instead of touching a scented thing: say "The brush of your fingers stirs loose a fresh cloud of [scent of the noun] smell."

Rule for printing the description of a dark room: try listening; try smelling; rule succeeds.

Instead of examining an audible thing while in darkness: try listening to the noun. Instead of examining something while in darkness: try touching the noun.

Before touching something when in darkness:
    say "You grope about..."

After inserting the plant into something:
    say "You unceremoniously dump [the noun] into [the second noun], hoping it sustains no important damage thereby."

Before printing the name of a dark room: if the player can touch an audible thing, say "Noisy "; if the player

can touch a scented thing, say "Perfumed ".

Visibility rule when in darkness:
    if examining something, there is sufficient light;
    there is insufficient light.

Rule for printing the announcement of darkness: say "It is now pleasantly lightless in here." instead.

Rule for deciding the scope of the player while in darkness: place the location in scope.

To decide whether in daylight:
    if in darkness, no;
    yes.

Instead of sleeping when in daylight:
    say "You've never been able to sleep with the light on."

Instead of sleeping when the player can touch an audible thing (called the irritant):
    say "The steady [sound of the irritant] from [the irritant] prevents your slumber."

Instead of sleeping when the player can touch a scented thing (called the irritant):
    if the irritant is chocolate, say "The smell of chocolate continues to tantalize you, keeping you from sleep.";
    otherwise say "You sniffle. [The irritant] is probably acting on your allergies."

```
Instead of sleeping:
    say "You slip easily into the arms of Morpheus.";
    end the story finally saying "At last..."


When play begins:
    say "You have at last escaped from the airport and
gotten through customs; survived an unnerving taxi ride
over icy highways; stared down the impertinent
concierge; endured the bellhop's catalog of features in
your room; and achieved, finally, a moment of peace.
Time for a good night's slumber!"


Test me with "listen / x dreamcatcher / switch lamp off /
look / sleep / eat chocolate / sleep / get plant / examine
plant / open suitcase / put plant in suitcase / close
suitcase / sleep / look / examine bose / switch bose off /
sleep".
```

Of course, this new version is less happy for the player, as we haven't included any way to silence the cat.

 ExampleThe Art of Noise *WI*

Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

This example involves redesigning the LISTEN command, removing its built-in function and replacing that with something more ambitious. We will learn more about how to do this later on.

```
"The Art of Noise"


A thing has some text called sound. The sound of a
thing is usually "silence".
```

The report listening rule is not listed in the report listening to rules.

Carry out listening to something:
    say "From [the noun] you hear [the sound of the noun]."

Instead of listening to a room:
    if an audible thing can be touched by the player, say "You hear [the list of audible things which can be touched by the player].";
    otherwise say "Nothing of note."

Definition: a thing is audible if the sound of it is not "silence".

Before printing the name of something audible while listening to a room:
    say "[sound] from the "

The Sharper Image is a room. The pet rock is a thing in the Sharper Image. The toy car is a thing in the Sharper Image. The sound of the car is "whirring and zooming". The plastic widget is a thing in the Sharper Image. The sound of the plastic widget is "bleeps and bloops". The pointless gadget is a thing in Sharper Image. The sound of the pointless gadget is "buzzbuzzbuzz".

The soundproof case is a transparent openable container in the Sharper Image. It is closed and fixed in place.

Test me with "listen / listen to rock / listen to car / get all / open case / put all in case / listen / close case / listen /

listen to car".

---

131 **Example** **Weathering** *WI*

The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

"Weathering"

A cloud pattern is a kind of value. The cloud patterns are cumulus, altocumulus, cumulonimbus, stratus, cirrus, nimbus, nimbostratus.

The Mount Pisgah Station is a room. "The rocky peak of Mt. Pisgah (altitude 872m) is graced only by an automatic weather station. The clouds, close enough almost to touch, are [a random cloud pattern]. Temperature: [a random number from 7 to 17] degrees, barometric pressure: [950 + a random number from 0 to 15] millibars."

Test me with "look / look / look".

---

157 **Example** **Full Moon** *WI*

Random atmospheric events which last the duration of a scene.

"Full Moon"

Wolf Pursuit is a scene. Wolf Pursuit begins when play begins.

Every turn during Wolf Pursuit, say "[one of]A twig snaps behind you![or]The wind howls in your ears. [or]You feel chilly.[at random]".

When play begins:
    say "You have lost your spectacles, and the lamp, and can see barely further than the next tree. Roots keep trying to trip you, too..."

The Dark Forest is a room. "You are mostly aware that you are not as alone here as you would like, and that the ground is uneven."

Test me with "z / z / z / z".

Example **Night and Day** *WI*

Cycling through a sequence of scenes to represent day and night following one another during a game.

Suppose we want to have a sequence of nights and days in our game, with one scene to govern each daylight condition.

"Night and Day"

The sun is a backdrop. It is everywhere. The description is "Currently out of sight."

```
Night is a recurring scene. Night begins when play
begins. Night begins when Dusk ends. Night ends when
the time since Night began is 3 minutes.
```

Notice that our two conditions for the beginning of Night are not in conflict: it will be night-time when the game begins, and then night will also recur every time the Dusk scene ends.

```
When Night begins:
    say "The sun falls below the horizon and the
temperature drops abruptly to well below zero.";
    now the description of the sun is "Currently out of
sight."
```

```
Dawn is a recurring scene. Dawn begins when Night
ends. Dawn ends when the time since Dawn began is 1
minute.
```

```
When Dawn begins:
    say "The sun appears on the horizon.";
    now the description of the sun is "It is tiny and weak.".
```

```
Day is a recurring scene. Day begins when Dawn ends.
Day ends when the time since Day began is 3 minutes.
```

```
When Day begins:
    say "The sun is now properly up."
```

```
Dusk is a recurring scene. Dusk begins when Day ends.
Dusk ends when the time since Dusk began is 1 minute.
```

```
When Dusk begins:
    say "The sun has passed across the sky and is on
the verge of setting."
```

```
Cratered Landscape is a room. "The ground here is [if
Night is happening]dim silver, with the craters visible as
darker splotches[otherwise]the color of dried blood; here
and there it is also rippled by impact craters[end if]. The
horizon curves visibly."


Test me with "z / z / z / look / x sun / z / z / z / z / z /
z".
```

If we run this example and then have a look at the scenes index, we'll see that the cycle is listed through thus:

```
Night (recurring)
    Dawn (recurring)
        Day (recurring)
            Dusk (recurring)
                Night
```

with the second "Night" in italics, to indicate that it is a repetition of the same scene that has already been listed above.

ExampleTotality

WI

To schedule an eclipse of the sun, which involves a number of related events.

First we define the events, and then we create a phrase to schedule them:

```
"Totality"


At the time when the penumbra appears:
    say "The sunlight dies away to an eerie, brownish
penumbra."
```

```
At the time when the eclipse begins:
    say "The moon passes across the disc of the sun,
plunging the world into darkness.";
    now the Chamber is dark.

At the time when the eclipse ends:
    say "The moon passes away from the sun, and a
gloomy penumbral light returns.";
    now the Chamber is lighted.

At the time when normal sunlight returns:
    say "The sun shines once more bright in the sky, not
to be eclipsed again on this spot for another thirty-seven
years."

To schedule an eclipse for (totality - a time):
    the penumbra appears at two minutes before totality;
    the eclipse begins at totality;
    the eclipse ends at three minutes after totality;
    normal sunlight returns at five minutes after totality.
```

Now we make use of the new phrase:

```
When play begins, schedule an eclipse for 3:27 PM.

The Chamber is a room.

The time of day is 3:25 PM.

Test me with "z / z / z / z / z / z / z / z / z".
```

We shall see much more about creating phrases later. Their advantage is

that they enable a complicated sequence of operations to be given a meaningful name, and that they can be re-used many times as needed.

ExampleOrange Cones *WI*

Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

Because we can invent our own adjectives (see the chapter on Phrases), we can make the conditions for a backdrop as simple or as complicated as we like.

In this scenario, we want the player to be able to take, move, and drop orange traffic cones to seal off one street or another. So we create our own "accessible" adjective as follows:

```
Definition: a road is accessible if the orange cones are
not in it.
```

...and now

```
move the traffic backdrop to all accessible roads.
```

will tell the traffic backdrop where to appear.

```
"Orange Cones"

A road is a kind of room.

The traffic is a backdrop. It is not scenery. The initial
appearance is "Dense traffic snarls the streets, making
it difficult to cross even with the lights. Men on
motorbikes edge between the cars, and sometimes pull
up onto the sidewalks to go around." The description is
"It is more or less as usual for this time of day. It's a
```

```
wonder it ever dissipates, really."


When play begins:
    move the traffic backdrop to all accessible roads.


A line of orange cones are a thing.


Definition: a road is accessible if the orange cones are
not in it.


After dropping the orange cones in a road:
    say "With steely determination you begin to lay out
the orange cones, blocking access to this segment of
street. This produces honking and swearing -- but you
persevere.";
    update backdrop positions.


After taking the orange cones:
    say "You go around taking up the orange cones, and
within moments the traffic begins to flow into the street
again.";
    update backdrop positions.
```

That accomplishes everything we set out to do, but let's add a very simple puzzle to test it out with:

```
The Town Square is a road. North of the Town Square
is Candle Street. Candle Street is a road. East of Town
Square is Mortar Street. Mortar Street is a road.


The line of orange cones are in Mortar Street.
```

```
The player wears a reflective vest and a hard hat.


The manhole cover is a door. "[if location is
accessible]Under the cars in the middle of the street,
you can just make out the cover of the manhole you
need to get into.[otherwise]There's a promising manhole
in the middle of the street.[end if]". It is closed and
openable. It is below the Town Square and above the
Access Tunnel.


Instead of opening the manhole cover when the location
is an accessible road:
    say "You can't get anywhere near the manhole cover
with all these cars above it."


Instead of taking the orange cones in Town Square:
    if the manhole cover is closed or Town Square is
accessible, continue the action;
    otherwise say "You'd better not let the traffic back in
until you've closed the manhole. There'll be accidents
otherwise."


After going to the Access Tunnel:
    say "With a stealthy glance left and right, you lower
yourself into the access tunnel, thus accomplishing your
mission for Chapter 2. To continue your adventure, see
Chapter 3: The Vault of Peaquod.";
    end the story finally.
```

And finally, a couple of features from the Activities chapter to make the output more elegant:

```
Rule for writing a paragraph about the orange cones:
say "A line of orange cones holds back the traffic from
entering here."
```

```
Rule for printing the name of the orange cones when the
cones are carried by the player:
    say "stack of orange cones".


Test me with "x traffic / open manhole / n / x traffic / s / e
/ x traffic / get cones / look / x traffic / w / drop cones /
look / open manhole / take cones / d".
```

Example **Uptown Girls**

WI

A stream of random pedestrians who go by the player.

Suppose we have an urban space we want to populate with random passers-by. These should have a range of characteristics and not always be described in the same way; and once the player has noticed one, he should be able to look at her further, until another pedestrian crosses his path.

```
"Uptown Girls"


Riverside Drive is a room. "There's a pleasant late-
afternoon view of the Hudson, and a snap in the air, and
you would rather be here than anywhere."


Instead of going a direction, say "Oh, you know where
you're going; no need to deviate from the usual path."


Instead of waiting, say "You stroll along enjoying the
November crispness."
```

It gets a little annoying to have a random event occurring every single turn of play, so let's introduce some randomness to determine how often the message appears:

Every turn when a random chance of 1 in 3 succeeds:
    reset passerby;
    choose a random row in the Table of Atmospheric Events;
    say "[event entry][paragraph break]"

Table of Atmospheric Events
event
"Slowly [a passerby] strolls by, turning to look at you as she passes."
"Some [passerby] nearly bumps into you."
"You dodge to avoid [a passerby]."
"You weave around [a passerby], who has stalled to look into a window."
"There's a ruckus as one of the ubiquitous taxis nearly collides with [a passerby] crossing the street."
"[The passerby] beside you waves to a friend across the street."
"To your left, [a passerby] drops her purse, and swears as she retrieves it."

Hair color is a kind of value. A person has hair color. the hair colors are red-headed, brunette, blonde.

Height is a kind of value. A person has height. The heights are tall, medium-height, short.

Grooming is a kind of value. A person has grooming. The groomings are messy and tidy.

To reset passerby:
    now the hair color of the passerby is a random hair color;
    now the height of the passerby is a random height;
    now the grooming of the passerby is a random grooming.

```
The passerby is a woman in Riverside Drive. The
passerby is scenery. Understand "woman" or "lady" as
the passerby. The printed name of the passerby is "[one
of]woman[or]lady[purely at random]".
```

At this point we borrow some techniques from later to describe the woman with a random combination of characteristics, and to let the player refer to her by those traits:

```
Before printing the name of the passerby:
    if a random chance of 1 in 4 succeeds, say "[height] ";
    if a random chance of 1 in 4 succeeds, say
"[grooming] ";
    if a random chance of 1 in 4 succeeds, say "[hair
color] ".
```

```
Understand the hair color property as describing the
passerby. Understand the height property as describing
the passerby. Understand the grooming property as
describing the passerby.
```

If we also wanted each of those combinations to mean some more specifically-described woman:

```
Instead of examining a passerby:
    repeat through Table of Passerby Descriptions:
        if hair entry is the hair color of the passerby and
height entry is the height of the passerby and grooming
entry is the grooming of the passerby, say "[description
entry][paragraph break]".
```

```
Table of Passerby Descriptions

hair color    height    grooming    description
red           tall      straggly    "A tall woman with long red hippie-hair poking out of a ponytail
in straggles, and bent to hide how tall she is."
red           short     straggly    "A short woman with shingled hair."
blonde        short     neat        "A street urchin, and very young, with ginger hair and a
smudged nose and far too many freckles."
```

"A henna-haired career woman with henna-red hair."

"A redhead in the way that lollipops and fire trucks are red: not by nature but by art. The rest of her clothing is pretty ordinary, though."

"Thin and small in every sense, with chin-length red-hair. Even high heels do not bring her head much above your shoulder."

"A Marilyn-esque woman with dark hair, wearing something resembling a tent."

"An unremarkable woman with dark brown hair and the aura of needing a wash."

"There are mustard stains on the t-shirt of this short brown-haired woman. Estimated age ca. 40. Possibly homeless."

"A leggy brunette in business attire."

"Medium-height, brown-haired, generally nondescript."

"A small little dark-haired girl."

"A blonde of about thirteen who looks as though she has not yet figured out how to get her wardrobe to catch up with her rate of growth. Her t-shirt and her pants are too short."

"Black tights and the wall-o-hair look."

"One of those shocking platinum blonde types, with a tiger-patterned skirt. Reeally trashy."

"Blonde and severe, with perfectly straight hair falling to the middle of the back."

"A bleached, Marilyn-esque blonde."

"Precise in a blue-and-white striped suit and a boyish haircut."

Test me with "z / z / x passerby / z / z / x passerby".

Chapter 4: Time and Plot

Of Time; *§4.2. Scripted Scenes*; *§4.3. Event Scheduling*; *§4.4. Scene Changes*; *§4.5. Flashbacks*; *§4.6*

Contents of *The Inform Recipe Book*
Chapter 3: Place
Chapter 5: The Viewpoint Character
Indexes of the examples

§4.1. The Passage Of Time

A story that makes heavy use of time may want to give the player a hint that time is important - and an easy way to keep track of how it's going - by adding the current time to the status line, instead of the score. To do this, we would write

```
When play begins: change the right hand status line to "[time of day]".
```

All else being equal, time passes at a rate of one minute per turn. But this need not be so: we can imagine a story where turns take much less time, or much more; or a story in which the passage of time was sometimes suspended, or one in which different actions required different amounts of time to perform.

Situation Room provides a way to print 24-hour time, while **Zqlran Era 8** implements a completely new measurement of time, for a story set on an alien world.

Uptempo and **The Hang of Thursdays** speed up time's passage: turns take fifteen minutes in the former, or a quarter day in the latter.

Timeless makes certain actions instant, so that they don't count against the clock; this is sometimes useful in timed situations where the player needs to review the situation before going on with a tricky puzzle. **Endurance** systematically extends this idea to allow us to assign different durations to any action in the story. **The Big Sainsbury's** goes the opposite direction, and meticulously adds a minute to the clock for all implicit take actions, just so that the player isn't allowed to economize on moves.

An alternative approach to time is not to tell the player specifically what hour of the day it is at all, but to move from one general time period to another as it becomes appropriate - when the player has solved enough puzzles, or worked his way through

enough of the plot. To this end we might use scenes representing, say, Thursday afternoon and then Thursday evening; then our scene rules, rather than the clock, would determine when Thursday afternoon stopped and Thursday evening began:

```
Thursday afternoon is a scene. Thursday evening is a scene.

Thursday afternoon ends when the player carries the portfolio.

Thursday evening begins when Thursday afternoon ends.
When Thursday evening begins:
    say "The great clock over St. Margaret's begins to chime 6.";
```

Though this gives time a loose relation to the number of turns played, it feels surprisingly realistic: players tend to think of time in a story in terms of the number of *significant* moves they made, while the random wandering, taking inventory, and looking at room descriptions while stuck don't make as big an impression. So advancing the story clock alongside the player's puzzle solutions or plot progress can work just as well as any stricter calculation.

See **Passers-By, Weather and Astronomical Events** *for cycles of day and night scenes*

See **Waiting, Sleeping** *for commands to let the player wait until a specific time or for a specific number of minutes*

See **Clocks and Scientific Instruments** *for clocks that can be set to times and that have analog or digital read-outs*

See **Timed Input** *for discussion of extensions allowing real-time input*

Start of Chapter 4: Time and Plot

Back to Chapter 3: Place: §3.9. Passers-By, Weather and Astronomical Events

Onward to §4.2. Scripted Scenes

Example 142: **Situation Room** Printing the time of day in 24-hour time, as in military

situations.

Example 377: ★ **The Big Sainsbury's** Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

Example 394: ★ **Uptempo** Adjust time advancement so the game clock moves fifteen minutes each turn.

Example 409: ★ **Timeless** A set of actions which do not take any game time at all.

Example 410: ★★ **Endurance** Giving different actions a range of durations using a time allotment rulebook.

Example 183: ★★★ **The Hang of Thursdays** Turns take a quarter day each, and the game rotates through the days of the week.

Example 260: ★★★ **Zqlran Era 8** Creating an alternative system of time for our game, using new units.

---

§4.2. Scripted Scenes

Sometimes we want to arrange a scene in which something goes on in the background (as though it were a movie playing) while the player goes about his business; or where a series of things has to happen before the player gets to the end.

The simplest way to arrange background events for a scene is to write the sequence of events into a table and work our way through it, printing one line per turn, until the scene runs out. **Day One** does exactly this.

At other times, we want a scene to last as long as it takes the *player* to do something. **Entrapment** lets the player poke around and explore as much as he likes, but ends as soon as he has accomplished the scene's goal - which, unfortunately for him, is to

get into an embarrassing situation so that another character can walk in and make fun of him. **The Prague Job** has a scene that requires the player to do a more specific set of tasks, but nags him and hurries him along until he's done.

Bowler Hats and Baby Geese assumes that our story is going to be assembled with a number of scenes, some of which will need to prevent the player from leaving the location until the scene is complete: it thus defines a "restricted" property for scenes, so that all such elements of the plot will work in the same way.

For more complex sorts of scripts and schedules, it may be worth consulting the extensions.

See **Characters Following a Script** *for a character whose conversation with the player is scripted to follow a pattern and then conclude*

Start of Chapter 4: Time and Plot

Back to §4.1. The Passage Of Time

Onward to §4.3. Event Scheduling

Example 162: ★ **The Prague Job** Scenes used to provide pacing while the player goes through his possessions.

Example 155: ★★ **Entrapment** A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing situation.

Example 159: ★★★ **Bowler Hats and Baby Geese** Creating a category of scenes that restrict the player's behavior.

Example 160: ★★★ **Day One** A scene which plays through a series of events in order, then ends when the list of events is exhausted.

---

§4.3. Event Scheduling

We can use a schedule of events to give some life to our environment: if we have a town setting, for instance, it makes sense for shops and libraries to open and close at set times; this is just what we find in **IPA**.

Air Conditioning Is Standard has characters who follow a timed schedule of events to interact with each other, while the player mostly wanders around missing out on the action. (Sometimes life is like that.) The same effects could have been achieved with scenes instead of clock times, but there are occasions when we do want to plan our characters' behavior to the minute rather than waiting for the player to be in the right place to observe it: in a murder mystery or a time-travel story, the exact timings might be quite significant.

We may also want to add events to the schedule during play, as in

```
Instead of pushing the egg-timer: say "It begins to mark time."; the
egg-timer clucks in four turns from now.

At the time when the egg-timer clucks: say "Cluck! Cluck! Cluck! says
the egg-timer."
```

Similarly, we can schedule things during play to happen at a specific time of day, as shown in **Hour of the Wren**.

See **Scene Changes** *for more things that arrive at pre-determined times*

See **Ships, Trains and Elevators** *for a train that follows a schedule, carrying the player along if he is aboard*

Start of Chapter 4: Time and Plot

Back to §4.2. Scripted Scenes

Onward to §4.4. Scene Changes

Example 141: **IPA** Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out

if he overstays his welcome.

Example 146: ★★★ **Hour of the Wren** Allowing the player to make an appointment, which is then kept.

Example 354: ★★★★ **Air Conditioning is Standard** Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

---

§4.4. Scene Changes

In a plot that takes place over multiple locations or has several distinct scenes, we may want to move the player or change the scenery around him. **Age of Steam** brings a train on and off-stage as the plot requires. **Meteoric** similarly brings a meteor into view at a certain time of day, showing off several implementations depending on whether or not the player is supposed to be able to refer to the meteor after it has gone.

Entrevaux constructs an organized system such that all scenes have their own lists of props and associated locations, and props are moved on and off automatically. Scene changes are also announced with a pause and a new title, such as "Chapter 2: Abduction".

Space Patrol - Stranded on Jupiter inserts an interlude in which the player's possessions and clothes are switched for new ones and the player moved to a new location - and then put back where he started from.

See **Flashbacks** *for more ways to move the player from one level of reality to another*

Start of Chapter 4: Time and Plot

Back to §4.3. Event Scheduling

Onward to §4.5. Flashbacks

Example 156: ★ **Age of Steam** The railway-station examples so far put together into a short game called "Age of Steam".

Example 119: ★★ **Meteoric I and II** A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

Example 158: ★★ **Space Patrol - Stranded on Jupiter!** We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

Example 163: ★★★ **Entrevaux** Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

---

§4.5. Flashbacks

The viewpoint character may often need to remember events long past. The easiest way to do this is with a cut-scene, in which at some relevant point we pause the story and print a long passage of text describing the memory. Because large amounts of text can be hard for the player to take in, we may want to include some pauses in the presentation of this material; this facility is provided by the Basic Screen Effects extension by Emily Short, and might work something like this:

```
Include Basic Screen Effects by Emily Short.

Instead of examining the photograph for the first time:
    say "This reminds you of the summer of '69...";
    wait for any key;
    say "... flashback content...";
```

wait for any key.

The "pause the game" phrase in the same extension offers a more dramatic pause that also clears the screen before printing new text.

Cut-scenes are easy to implement but should be used sparingly, since players often get impatient with long uninteractive passages. A slightly more deluxe implementation might insert an interactive scene that simply happens to be set in the past, before going on with another scene set "now"; and, indeed, some IF abandons the idea of "now" entirely, presenting pieces in a non-chronological order and letting the player work out how the sequence works together.

The most challenging case to implement (though still not very hard) is the one where we remove the player from one scenario, let him play through a flashback with past possessions and clothing, and then restore him to the same situation he left, with all of the same possessions and clothing. **Pine 3** shows how to do this: the code to change the player's status is isolated at the end of the example, and might fruitfully be reused. **Pine 4** expands on the same idea by adding another flashback scene, demonstrating one that can be visited repeatedly and one that can be seen only once.

See **Scene Changes** *for more uses of stripping and restoring the player*

See **Background** *for other ways of introducing information that the player character already knows*

See **Alternate Default Messages** *for comments on how to change the tense of an interactive scene*

Start of Chapter 4: Time and Plot

Back to §4.4. Scene Changes

Onward to §4.6. Plot Management

Example 165: **Pine 3**
Pine: Allowing the player to visit aspects of the past in memory and describe these events to the princess, as a break from the marriage-proposal scene.

Example 167: **Pine 4**
Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the

Princess has understood the point.

§4.6. Plot Management

A plot manager (sometimes called a drama manager) is a piece of the program whose job it is to plan out events so that, whatever the player does, the story advances and an interesting narrative results. The plot manager might, for instance, decide that the player has wandered around for too many scenes without making any progress, and might compensate by making something happen that gives him a new hint on his current problem. It might trigger characters to act when it thinks the story should be reaching a crisis point. It might introduce new complications when it determines that the player is running out of problems to solve.

This is a theoretically challenging field. Sophisticated plot management requires that the story make difficult guesses, such as whether the player is "stuck" and what the player is working on right now. The advantage of using such a system is that (done very well) it makes the story extremely responsive to the player's behavior, which means that he is a real agent in the unwinding of the plot. It also contributes to the replayability, since trying the story a second or third time will produce quite different outcomes. But it is procedurally difficult to design a good plot management system and it requires a huge amount of content, as well: in order for the plot manager to give the player hints, change the course of events to suit his focus, and so on, the story has to have available many, many more scenes than will ever occur in any single playing.

Fate Steps In is only a *very* brief sketch in this direction, one in which the "fate" entity is trying to accomplish an end goal and, every turn, looks for ways to push the story towards that conclusion, whatever the player does.

See **Goal-Seeking Characters** *for alternate ways to make characters act on their own*

Start of Chapter 4: Time and Plot

Back to §4.5. Flashbacks

Onward to Chapter 5: The Viewpoint Character: §5.1. The Human Body

Example 209: **Fate Steps In** Fate entity which attempts to make things happen, by hook or by crook, including

taking preliminary actions to set the player up a bit.

Examples from Chapter 4: Time and Plot

Start of this chapter
Chapter 5: The Viewpoint Character
Indexes of the examples

142 Example Situation Room *WI*

Printing the time of day in 24-hour time, as in military situations.

Though Inform normally prints times in AM/PM terms, it stores the hours and minutes as 24-hour time; so, if we like, we can easily extract that information again thus:

```
"Situation Room"

The Situation Room is a room.

To say (relevant time - a time) as 24h time:
    let H be the hours part of relevant time;
    let M be the minutes part of relevant time;
    say "[if H is less than 10]0[end if][H][if M is less than
10]0[end if][M]".

When play begins:
    now the time of day is 6:09 PM;
    now the right hand status line is "[time of day as 24h
time]".
```

Test me with "z".

Example **The Big Sainsbury's**

Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

Implicit takes are a convenience to players; in general, we would like to avoid asking players to type any more obvious commands than strictly necessary, while allowing the computer to guess as much as it safely can.

Occasionally, though, we have designed a timed puzzle in which the player has a limited number of moves in which to accomplish his objectives. In that case, the implicit take complicates matters, because it means that a player who types

```
>EAT GATEAU
(first taking the gateau...)
```

gets away with a spare move compared to the precise but naïf dupe who types

```
>TAKE GATEAU
>EAT GATEAU
```

...and really, that doesn't seem quite fair. The way to fix this problem is to fill in the extra minute on the clock during the implicit take; and that is indeed what we do in the following example.

```
"The Big Sainsbury's"


Sainsbury's is a room.


The crispy duck and the Guinness steak pie are edible
```

things in Sainsbury's.

Rule for implicitly taking something:
    follow the advance time rule;
    continue the activity.

When play begins:
    now the right hand status line is "[time of day]".

Test me with "take crispy duck / eat crispy duck / eat steak pie".

ExampleUptempo

Adjust time advancement so the game clock moves fifteen minutes each turn.

Suppose a game in which all actions take a very long time. Here's a simple implementation:

"Uptempo"

The fast time rule is listed instead of the advance time rule in the turn sequence rules.

This is the fast time rule:
    increment the turn count;
    increase the time of day by 15 minutes.

When play begins: now the right hand status line is

```
"[time of day]".


The Temporal Hot Spot is a room.


Test me with "z / z".
```

This works fine as it stands, but we may run into some difficulty with it if we add scheduled events:

```
At 9:30 AM:
    say "Two turtles run by, almost too fast to see."


At 9:37 AM:
    say "A snail blitzes past."


At 9:42 AM:
    say "The grass grows."


At 9:50 AM:
    say "Several flowers burst open."
```

Time is counted forward after the schedule has already been consulted, so that only the 9:30 AM event happens between 9:30 and 9:45; the next two appear to occur between 9:45 and 10:00 AM, and the 9:50 AM event is not reported until the 10:00 AM to 10:15 wait. To get around this, we might schedule events only on the fifteen-minute mark when we want them to occur. Alternatively, we might try instead

```
"Uptempo"


The fast time rule is listed before the timed events rule
```

```
in the turn sequence rules.


The advance time rule is not listed in the turn sequence
rules.


This is the fast time rule:
    increment the turn count;
    increase the time of day by 15 minutes.


When play begins: now the right hand status line is
"[time of day]".


The Temporal Hot Spot is a room.


At 9:30 AM:
    say "Two turtles run by, almost too fast to see."


At 9:37 AM:
    say "A snail blitzes past."


At 9:42 AM:
    say "The grass grows."


At 9:50 AM:
    say "Several flowers burst open."


Test me with "z / z / z / z".
```

This time our revised time-advancing rule is listed just before the event

scheduler, not just afterwards.

---

409 **ExampleTimeless**
A set of actions which do not take any game time at all.

WI

In a game with tight timing, it is sometimes friendliest to the player to let him LOOK and EXAMINE as much as necessary without being penalized.

```
"Timeless"

Examining something is acting fast. Looking is acting
fast.
```

Now we need a rule which, just at the right moment, stops the turn sequence rulebook in the cast of our new fast-acting actions:

```
The take visual actions out of world rule is listed before
the every turn stage rule in the turn sequence rules.

This is the take visual actions out of world rule: if acting
fast, rule succeeds.
```

Thus the rest of the turn sequence rulebook is omitted for looking or examining: in effect, they become out-of-world actions like "saving the game". If we wanted to add, say, taking inventory to the list of instant activities, we would just need to define it as acting fast, too.

Now the scenario for testing:

```
When play begins:
    say "You are cornered by a pack of zombie wolves,
armed only with a torch and a pair of pinking shears.
```

This may be your last moment on earth, unless you can think fast!"

Cleft is a room. "You're backed into a cleft in the granite: behind you are only steep, high faces of stone, and before you a narrow passage."

The plural of zombie wolf is zombie wolves. A zombie wolf is a kind of animal. Four zombie wolves are in Cleft.

Rule for writing a paragraph about zombie wolves:
    say "The good news is that there isn't much space in which for the zombie wolves to attack.";
    now every zombie wolf is mentioned.

A steep high face of stone is scenery in Cleft. Understand "rock" as the stone. The description is "Now that you look more closely, there appear to be pitons driven into the rock."

Some pitons are part of the stone. The description of the pitons is "It looks as though someone else has made this ascent before."

Instead of climbing the stone, try going up. Instead of climbing the pitons, try going up.

Above the Cleft is Clifftop.

Every turn when the location is Cleft:
    say "Alas, your time has run out. The alpha wolf springs--";

```
    end the story.


Every turn when the location is Clifftop:
    say "After a breathless climb, you emerge at last onto
the open clifftop.";
    end the story finally.


Test me with "x me / x stone / x pitons / climb pitons".
```

ExampleEndurance

Giving different actions a range of durations using a time allotment rulebook.

Here we move to a systematic way of giving different durations to different actions, including even variations on the same act -- so that for instance climbing a steep hill might take several minutes more than other going actions. We do this by setting a number, "work duration", to represent the number of minutes consumed by a given action, and then consulting a rulebook to find out how long the past turn's action should take. By default, an action will take 1 minute.

We'll start by emulating the behavior of "Uptempo": each turn we'll set the clock forward most of the way, then check to see what has changed since the last turn, print any relevant events, and only then set the clock forward the final minute. The exception is when an action is set to take no time at all; in that case, we'll skip the rest of the turn sequence rules entirely.

```
"Endurance"


Work duration is a number that varies.


Every turn:
    now work duration is 0;
```

increment the turn count;
follow the time allotment rules;
if work duration is 0, rule succeeds;
increase the time of day by (work duration minutes - 1 minute).

The time allotment rules are a rulebook.

A time allotment rule for examining or looking:
now work duration is 0;
rule succeeds.

A time allotment rule for going:
now work duration is 2;
rule succeeds.

A time allotment rule for going up:
now work duration is 5;
rule succeeds.

A time allotment rule for waiting:
now work duration is 10;
rule succeeds.

The last time allotment rule:
now work duration is 1.

When play begins: now the right hand status line is "[time of day]".

The Quai is a room. "An attractive park at the edge of the river Aude: here you can wander among palm trees,

and watch cyclists go by on the bike path; in the water there are ducks. In the cafe to your north, patrons sip their pastis; and above you is the medieval walled city and its castle."

The Cafe is north of the Quai. "A charming collection of umbrella-shaded tables, from which one can watch the river and the walls of the city beyond. The noise of traffic is only a minor distraction."

The City is above the Quai.

After going to the City:
    say "You struggle uphill for some distance...";
    continue the action.

At 9:15 AM:
    say "The bells ring out from Place Carnot."

Test me with "z / n / s / u".

Example **The Hang of Thursdays**

Turns take a quarter day each, and the game rotates through the days of the week.

"The Hang of Thursdays"

The Stage is a room. Rule for printing the name of the stage: say "[current weekday] [current time period]"

instead.

A weekday is a kind of value. The weekdays are Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday. The current weekday is a weekday that varies. The current weekday is Saturday.

A time period is a kind of value. The time periods are morning, afternoon, evening, night. The current time period is a time period that varies. The current time period is afternoon.

This is the new advance time rule:
    if the current time period is less than night:
        now the current time period is the time period after the current time period;
    otherwise:
        now the current time period is morning;
        now the current weekday is the weekday after the current weekday.

Now we need to borrow from a later chapter to make these instructions apply to the passage of time:

The new advance time rule is listed instead of the advance time rule in the turn sequence rules.

Test me with "z / z / z / z / z".

ExampleZqlran Era 8
Creating an alternative system of time for our game, using new units.

Suppose that our game takes place on an alien planet that does not

follow Earth time. On this planet, we want to track time with different units. We also want time to advance in those units, and we want to be able to set a schedule of timed events.

```
"Zqlran Era 8"


The Barren Lavender Surface of Zql is a room. "It is late
twilight on Zql. Overhead, two crescent moons, both
green, mark the sluggish passage of time. A cold wind is
blowing over the pale purplish ground cover, but it does
not penetrate your airtight suit."


A Zqlran date is a kind of value. 14-88 specifies a Zqlran
date with parts zqls and frbs. Current zqlran date is a
zqlran date that varies. The current zqlran date is 8-22.
Previous zqlran date is a zqlran date that varies. The
previous zqlran date is 8-20.


When play begins:
    now left hand status line is "[current zqlran date], or
[current zqlran date in words]".


To say (Zqlra - a Zqlran date) in words:
    say "[zqls part of Zqlra] Z, [frbs part of Zqlra] f."
```

Inform automatically supplies a way to say a new unit, which will look similar to the format in which we defined that unit in the first place. But we can (as shown here) create our own alternative say phrases to express the units in other ways as well.

Next, we need to meddle with time advancement so that time is tracked in Zqlran date rather than in minutes. This requires borrowing a trick from a later chapter, to replace Inform's built-in time handling with an alternative time handling rule of our own:

```
The Zqlran time rule is listed instead of the advance
```

time rule in the turn sequence rules.

```
This is the Zqlran time rule:
    increment turn count;
    now the previous zqlran date is current zqlran date;
    increase the current zqlran date by 0-02;
    repeat through the Table of Zql Schedule:
        if era entry is greater than previous zqlran date and
era entry is not greater than current zqlran date:
            say event entry;
            say paragraph break;
            blank out the whole row.
```

```
Table of Zql Schedule

era    event
0-24   "A wisp-thin cloud blows rapidly across the face of Nepenthe, the
lesser of the two green moons."
0-28   "The cloud across Nepenthe clears."
```

Note that we could if we wished use a different device for scheduling events: this one simply prints text at scheduled eras, but we might also (for instance) make the event entry be a rule for Inform to follow, and tell Inform to carry out that rule at the scheduled time.

Example **The Prague Job**

Scenes used to provide pacing while the player goes through his possessions.

Suppose we want to remind the player that he doesn't have all the time in the world, by starting to nag him when he's nearly, but not entirely, done going over his inventory in preparation for a job.

```
"The Prague Job"

A thing can be seen or unseen. A thing is usually
```

unseen. Carry out examining: now the noun is seen.

The player carries a lockpick, a smoke bomb, a grappling hook, and a pair of gloves. The description of the lockpick is "Effective on most kinds of key locks, it is a gift from your mentor in the discipline, old Wheezy." The description of the smoke bomb is "Your last of these, so you should rely on it only when other modes of escape have vanished. It takes effect when dropped, producing a cloud of purple haze sufficient to fill a medium-sized room." The description of the grappling hook is "Good for shooting at balconies and other sorts of overhang." The gloves are wearable. The description of the gloves is "Black and shiny, with gripping material on the palms. Batman would be jealous."

The Toilet is a room. "The walls are painted an unattractive green; the fixtures are a bit old. But it is the only place in the hostel with any privacy." The Long Hallway is outside from the Toilet.

Reviewing Possessions is a scene. Reviewing Possessions begins when play begins.

Escalating Danger is a scene. Reviewing Possessions ends when Escalating Danger begins. Escalating Danger begins when preparations near completion.

To decide whether preparations near completion:
    if at least two of the things which are carried by the player are seen, yes;
    no.

When Escalating Danger begins: say "Someone pounds on the door of your hideout and yells at you in Czech."

Instead of going from the Toilet during Reviewing Possessions: say "You need to go over your equipment first, and make sure you're ready here."

Instead of going from the Toilet during Escalating Danger: say "You're not done checking over your materials."

Instead of waiting during Escalating Danger: say "There's no time to waste."

Every turn during Escalating Danger: if the time since Escalating Danger began is greater than 1 minute, say "Impatient footsteps pass your door again."

Escalating Danger ends when every thing which is carried by the player is seen. When Escalating Danger ends, say "There -- nothing damaged or torn. You're ready to go."

Mission is a scene. Mission begins when the player is in the Long Hallway. When Mission begins: end the story saying "The game is afoot"

Test me with "i / x lockpick / out / x bomb / out / x hook / x gloves / out".

A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing

situation.

The power of scenes lies in their ability to watch for general conditions and move the narrative along whenever these are fulfilled. Instead of waiting for the player to do one specific thing, the game waits for the world to be in a certain condition, before moving to the next stage of the plot.

For instance, suppose we have a story in which the player has been captured for doing something inappropriate at court and is brought in to await a meeting with a palace official. We want to give the player a few minutes to stew, and we want the scene to end with him doing something mildly peculiar or embarrassing, and the official catching him in the act. So we tempt him into trying any of a number of different kooky activities, and just wait until he falls into the trap...

"Entrapment"

Waiting Suite is a room. "You find yourself in a narrow room, more cozy than is really comfortable, with dark paneling on all the walls. Underfoot is a thick carpet the color of dried blood. The head of a dragon kit is mounted on the wall."

The wood paneling is scenery in the Waiting Suite. The description is "Just the sort of ornate panels that might conceal a carved switch. You've heard all sorts of rumors about secret rooms and passages in the palace, some of which have not been opened in centuries because no one remembers how to get at them." Understand "panels" or "panel" or "panelling" as the paneling.

Instead of switching on the paneling, say "First you'll have to locate any switches or catches with a careful search."

The thick carpet is scenery in the Waiting Suite.

Understand "red" or "blood" or "rug" as the carpet. The description is "A dull, unwelcoming weave, only a touch redder than the wood around you. You discern that it does not lie perfectly flat."

Instead of touching the paneling for the first time: say "You run your hands over the paneling with a methodical touch, knowing exactly what you're looking for but never quite feeling anything that gives or twists; then thump lightly, looking for hollow spaces."

Instead of touching the paneling for the second time: say "With increased vigor, you run your fingers along the borders between panels, then smack each panel sharply at the center. No luck yet, but if you keep at it, you're bound to turn up anything that's there to find."

Instead of attacking the paneling: try touching the paneling. Instead of searching the paneling: try touching the paneling. Understand "knock on [something]" or "tap [something]" or "tap on [something]" as attacking.

After touching the paneling when the player is not confident:
    say "Having polished off all the panels within easy reach, you now have to contort yourself around furniture here and crawl along the floorboards there, hitting each panel three times quite solidly before moving on.";
now the player is embarrassed.

Instead of looking under the carpet for the first time:
    say "You take a corner of the carpet and tug. The floor is sticky, so it doesn't come up on the first try."

A small table is an enterable supporter in the Waiting Suite. On the table is a copy of Dragon Pursuit Today.

The description of Dragon Pursuit Today is "Full of glossy illustrations of dragons in various stages of capture, captivity, and destruction. The back of the magazine contains small black-and-white advertisements for hunting kits and the like." Some advertisements and some illustrations are part of Dragon Pursuit Today. The description of the illustrations is "You have the misfortune to look first at the photographs accompanying 'Cleaning Dragon Splanchna', and feel quite unwell." The description of the advertisements is "Mostly terse ads and phone numbers."

After looking under the table:
    say "It's quite a low table and you have to get down on your knees and poke your head underneath in order to get a good look."

After looking under the carpet:
    say "You pull again at the carpet. There is a tug, then a tearing, as the ancient fabric struggles against the fabric glue. Some of the carpet winds up in your hand and some of it remains in patchy threads adhering to the floor."

After entering the table:
    say "You climb onto the small table, noticing belatedly that you are leaving muddy footprints on its polished surface. Oh well: you can wipe them away again when you get down."

The dragon head is scenery in the Waiting Suite. Understand "kit" or "mouth" as the dragon head. The description is "Its eyes are wide with bewildered surprise; its mouth gapes, its forked tongue protrudes indignantly. From down here it looks as though there's something shiny stuck in its mouth, though you can't tell for sure." The head contains a shiny thing. The description of the shiny thing is "Intriguing but impossible to see clearly." Instead of taking the shiny thing, try searching the dragon head.

Before searching the dragon head:
    if the player is not on the table, try entering the table;
    if the player is not on the table, stop the action.

After searching the dragon head: say "You have a good look inside the dragon's mouth. There's a ball of lucite inside, propping the jaw in display position."

A person can be confident, nervous, or embarrassed. The player is confident.

Touching the paneling is embarrassing behavior. Looking under the carpet is embarrassing behavior. Entering the table is embarrassing behavior. Looking under the table is embarrassing behavior.

Instead of embarrassing behavior:
    if the player is nervous, now the player is embarrassed;
    if the player is confident:
        say "Before you can act, you hear movement from the inner office. You freeze, not quite ready to be discovered in this situation. But no one comes out, and you begin to breathe more easily.";
        now the player is nervous;
    otherwise:
        continue the action.

Causing trouble is a scene. Causing trouble begins when play begins. Causing trouble ends when the player is embarrassed. When Causing trouble ends: say "Just at this inopportune moment, you hear a throat being cleared behind you. 'We can see you now within,' says a dry voice."; end the story saying "To be continued..."

Test me with "switch paneling / touch paneling / g / g / g".

Test more with "x dragon / x shiny / search head / g".

...and this scene might lead to another, and so on.

The purpose of an open-ended scene like this might be puzzly or narrative: we might be waiting for the player to get a puzzle solved, or we might be waiting for him to fulfil some plot condition that must be met before we can go on.

ExampleBowler Hats and Baby Geese

Creating a category of scenes that restrict the player's behavior.

Scenes can have properties -- a fact that is very useful when it comes to writing a series of scenes that all need to act alike in some respect.

Suppose we have a plot that features a number of scripted scenes, where we need the player to stand still and wait while the events of the scene play out. One way to set this up is to create a property for such scenes -- let's call them "restricted" -- and then write a rule that keeps the player in place while the scene happens:

"Bowler Hats and Baby Geese"

Section 1 - The Procedure

A scene can be restricted or free.

Instead of going somewhere during a restricted scene:

```
    say "Better to stay here for the moment and find out
what is going to happen next."
```

And now let's set up our restricted scene. In it, a clown is going to turn up wherever the player is (it doesn't matter where on the map he's gotten to at this point) and do a performance; the player will not be able to leave the area until the performance completes. We'll start with the setting:

Section 2 - The Stage and Props

```
The Broad Lawn is a room. "A sort of fun fair has been
set up on this broad lawn, with the House as a
backdrop: it's an attempt to give local children
something to do during the bank holiday. In typical
fashion, everyone is doing a very good job of ignoring
the House itself, despite its swarthy roofline and dozens
of blacked-out windows."
```

```
The House is scenery in the Broad Lawn. The
description is "A cautious vagueness about the nature of
the inhabitants is generally considered a good idea.
They might be gods, or minor demons, or they might be
aliens from space, or possibly they are embodiments of
physical principles, or expressions of universal human
experience, or... at any rate they can run time backward
and forward so it warbles like an old cassette. And
they're always about when somebody dies. Other than
that, they're very good neighbors and no one has a
word to say against."
```

```
Instead of entering the House:
    say "You can't go in, of course. It's not a house for
people."
```

```
The Gazebo is north of the Broad Lawn. "The gazebo is
sometimes used for bands, but at the moment has been
appropriated for the distribution of lemonade."
```

The clown is a man. "A clown wearing [a list of things worn by the clown] stands nearby." The description is "He winks back at you."

The clown wears a purple polka-dot bowler hat. He carries a supply of baby geese. The description of the supply of baby geese is "Three or four. Or five. It's hard to count." Understand "goose" or "gosling" or "goslings" as the supply of baby geese.

There are some eggs. The description of the eggs is "A blur, really."

There is a Spanish omelet. The description of the Spanish omelet is "Exquisitely prepared."

...And now the scene itself:

Section 3 - The Scenes

The Clown Performance is a restricted scene. Clown Performance begins when the turn count is 3.

When Clown Performance begins:
    move the clown to the location.

Every turn during Clown Performance:
    repeat through the Table of Clowning:
        say "[event description entry][paragraph break]";
        blank out the whole row;
        stop.

When Clown Performance ends:
    now the eggs are nowhere;
    now the clown carries the omelet.

Clown Performance ends when the number of filled rows in the Table of Clowning is 0.

Table of Clowning
event description
"A clown with a purple polka-dot bowler hat strides into the vicinity and begins to juggle baby geese."
"While the clown juggles, the baby geese visibly grow older and larger. The clown becomes unnerved."
"In an attempt to resolve the problem, the clown reverses the direction of his juggling. The geese revert to goslings."
"The goslings become smaller and smaller until the clown is juggling goose eggs[replace eggs]."
"The clown throws all the eggs into the air at once and catches them in the bowler hat. He takes a bow; the audience applauds. As a final gesture, he upends his hat to release a perfectly cooked omelet."

To say replace eggs:
    now the supply of baby geese is nowhere;
    now the clown carries the eggs.

Free Time is a scene. Free Time begins when Clown Performance Ends.

Test me with "scenes / n / z/ z / look / x geese / s / x geese / x eggs / z / s".

A scene which plays through a series of events in order, then ends when the list of events is exhausted.

```
"Day One"

Lecture is a scene. Lecture begins when play begins.

Every turn during Lecture:
    repeat through Table of Lecture Events:
        say "[event entry][paragraph break]";
        blank out the whole row;
        rule succeeds.
```

Here we use a table (see subsequent chapters) to keep track of all the events we wish to have occur during the course of the scene.

```
Table of Lecture Events
event
"'Welcome to Precolumbian Archaeology 101,' thunders Dr Freitag from the front of the class. 'Miss-- yes, you in the back. If you can't find a free seat, how are you going to find Atlantis? Sit down or leave. Now. Thank you.'"
"Freitag stands behinds his desk and lines up the pile of books there more neatly. 'It has come to my attention over previous years that there are two sorts of person who enroll in my class,' he says.

'Some of you will be members of the swim team or women's lacrosse players who have a distribution requirement to fulfill and are under the mistaken impression that archaeology must be easier than psychology. If that description applies to you, I advise you to drop the class now rather than at the midterm break. Under absolutely no circumstances will I ever sign a withdrawal form for someone who is crying at the time. Make a note of that, please.'"
"'The second sort of person,' Dr Freitag says, getting
```

another wind. 'Yes, the second sort of person takes this class because she imagines that it is going to lead to adventure or possibly to new age encounters with dolphins.'

His eye moves over the class, lingering an especially long time on a girl in a patchwork skirt.

'You should also leave now, but since you are probably lying to yourself about the reasons you're here, you will probably not heed my warning and we will be doomed to a semester of one another's company nonetheless.'"
"'Whatever you may tell yourself, you are not here to gain a deeper understanding of the world or get in touch with yourself or experience another culture.'

He paces before the first row of desks, hammering on them one at a time. 'I know you probably wrote an admissions statement saying that that is what you hoped to do. Well, too bad. It is not inconceivable that some of you, somehow, will muddle towards a deeper understanding of something thanks to this class, but I am not holding my breath, and neither should you.'"
"Freitag takes a breath. 'No, my dear freshwomen, what you are here to do is learn facts. FACTS. Facts are unpopular in this university and, I am unhappily aware, at most of the institutions of inferior preparation from which you have come. Nonetheless, facts it will be. I will expect you to learn names. I will expect you to learn dates. I will expect you to study maps and I will expect you to produce evidence of exacting geographical knowledge on the exams. I will expect you to learn shapes of pottery and memorize masonry designs. There are no principles you can learn which are more important or more useful than a truly colossal bank of facts right there in your own head.'"
"'I do not ever want to hear that you do not need to learn things because you will be able to look them up. This is the greatest fallacy of your computer-semi-literate generation, that you can get anything out of Google if you need it. Not only is this demonstrably false, but it overlooks something phenomenally important: you only know to look for something if you already know it EXISTS. In short there is no way to fake knowledge,

```
and I am not going to pretend there is.' He smiles in
lupine fashion.
```

```
'This class is likely to be the most miserable experience
of your four years in university. Clear?'"
"Everyone is silent."
"The lecture is interrupted by the shrill of a bell."
```

And then we define the scene so that it ends when the table runs out.

```
Lecture ends when the number of filled rows in the
Table of Lecture Events is 0.
```

One advantage of this is that we can then edit the events in the scene by changing just the table; the scene will always run the right length and end on the turn when the last event occurs.

And to add a few additional details:

```
Instead of doing something other than waiting, looking,
listening or examining during Lecture:
    say "Dr Freitag glares at you so fiercely that you are
frozen into inaction."
```

Notice the careful phrasing of "doing something other than..." so that we do not mention the objects; if we had written "something other than listening to something...", the instead rule would match only action patterns which involved a noun. We state the rule more generally so that it will also match nounless commands such as JUMP and SING, since Freitag will probably take a dim view of those as well.

```
When Lecture ends:
    now Freitag is nowhere;
    say "There is a flurry of movement as your fellow
students begin to put away their books. Dr Freitag
makes his way to the door and is gone before anyone
can ask him anything."
```

```
The Classroom is a room. Dr Freitag is a man in the
```

Classroom. "Dr Freitag paces before the blackboard."

Test me with "listen / x dr / x me / jump / z / z / z / z / z / x dr".

Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out if he overstays his welcome.

Suppose we wanted a game set in a living town, with locations opening and closing at different times of day, and business carrying on as usual. The point might be to force the player to plan his itinerary carefully to hit the right spots at the right times; or we might be writing a more contemplative piece, where part of the enjoyment came from just watching the characters wander around doing their daily business...

"IPA"

When play begins: now the right hand status line is "[time of day]".

The time of day is 9:50 AM.

A shop is a kind of room. A shop has a time called the opening hour. The opening hour of the shop is usually 8 AM. A shop has a time called the closing hour. A shop usually has closing hour 6 PM.

Check going to a shop (called the target):
    if the time of day is before the opening hour of the target,

say "[The target] is locked up until [the opening hour of the target]." instead.

Check going to a shop (called the target):
if the time of day is after the closing hour of the target,
say "[The target] has been closed since [the closing hour of the target]." instead.

Every turn when the location is a shop:
let deadline be the closing hour of the location;
if the deadline is before the time of day:
let target be a random adjacent room which is not a shop;
say "You are gently but firmly ushered out, since [the location] is closing.";
move the player to the target.

The Strip-mall Parking Lot is a room. "Dead Christmas trees are heaped outside the bagel shop. Strips of dirty ice survive along the curb, and in the shadows of the lamp-posts. A wet, almost illegible sheet of algebra homework is plastered to the asphalt.

Pinewood Brewing Supply is at the east end of the lot."

Pinewood Brewing Supply is a shop. It is east of Parking Lot. The opening hour of Pinewood Brewing Supply is 10:00 AM. The closing hour of Pinewood Brewing Supply is 3:30 PM. "Shelves and shelves of malt and hops; large glass carboys, and plastic tubing; empty bottles; bottle-caps; bottle-labeling kits; starters for vinegar, sourdough, root beer.

A sweet malty smell hangs in the air."

Instead of going to Brewing Supply when the time of day is before the opening hour of Brewing Supply for the second time:
    say "You rattle at the door again. 'Hold your horses, for crying out loud,' yells a voice from within."

Noah's Bagels is a shop. It is north of the Parking Lot. The opening hour of Noah's Bagels is 6:00 AM. The closing hour of Noah's Bagels is 11:00 AM. "The selection has been somewhat picked over, leaving you with your choice of Pumpernickel, Asiago, or Everything."

Test me with "e / e / n / z / s / e / z / e / z / z / e".

ExampleHour of the Wren

Allowing the player to make an appointment, which is then kept.

Here we allow the player to set the time at which some future event is going to happen, rather than letting the game decide. We'll need to borrow the syntax for defining new actions from a later chapter:

"Hour of the Wren"

When play begins:
    say "You more or less stumble across them in Central Park: a disparate group of people, all of different ages, sitting in a circle. They aren't talking to one another -- in fact, they seem to be trying very hard to ignore one another, like people in the waiting room of an especially embarrassing kind of doctor. You are about to go around when a woman in a grey pressed suit comes up to you. Her suit-skirt is trimmed in lavender cord, and she looks as though she might have been extremely sharp-dressed in 1944. She hands you a card."

The Circle in the Grass is a room. "No one is looking at you, except for the secretary, if that is what she is."

The player carries a card. The description of the card is "Typed: 'Active astrology - dislike your fortunes? change your stars! - make an appointment now - hour of the wren STILL AVAILABLE.'".

The time of day is 1:55 PM.

Understand "pick [time]" or "choose [time]" or "make appointment for [time]" or "make an appointment for [time]" as making an appointment for. Making an appointment for is an action applying to one time.

Carry out making an appointment for:
    say "Fate cannot be commanded more than once."

Instead of making an appointment for the time understood for the first time:
    say "You settle on [the time understood] for your appointment. The woman makes a note of it in an appointment book, which she carries in a brown paper bag. 'Excellent choice, ma'am,' she says in a low, urgent voice. 'You'll be very satisfied.'";
    stars shift at the time understood.

Understand "hour of the wren" as 2:00 PM.

At the time when stars shift:
    end the story saying "insert cataclysm here".

Test me with "x card / make appointment for hour of the wren / z / z / z / z".

ExampleAir Conditioning is Standard *WI*

Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

"Air Conditioning is Standard"

Section 1 - The Garage

A person has some text called current occupation. The current occupation of a person is usually "None".

Mood is a kind of value. The moods are bemused, bored, attentive, rapt, and blushing. A person has a mood. A person is usually attentive.

Instead of examining a person:
    now every thing is unmentioned;
    carry out the writing a paragraph about activity with the noun.

Rule for writing a paragraph about a person (called X):
    let the subsequent mention be "Name";
    if the current occupation of X is not "None":
        say "[current occupation of X]. ";
        let the subsequent mention be "He";

```
        if X is female, let the subsequent mention be "She";
    if X wears something unmentioned:
        if the subsequent mention is "Name", say "[The X]
";
        otherwise say "[subsequent mention] ";
        say "is wearing [a list of unmentioned things worn
by X]";
        if X carries something unmentioned, say " and
carrying [a list of unmentioned things carried by X]";
        say ".";
    otherwise:
        if X carries something unmentioned:
            if the subsequent mention is "Name", say "[The
X] ";
            otherwise say "[subsequent mention] ";
            say " is carrying [a list of unmentioned things
carried by X]."
```

The Garage is a room. "Above the street door is a spectacular art nouveau fanlight, wherein a stained-glass Spirit of Progress bestows the gift of Transportation on mankind.

The sun, gleaming through the hair of Progress, throws amber curls on the macadam floor."

The fanlight is scenery in the Garage. The description is "A semi-circle of stained glass as wide as the garage door, designed by Louis Comfort Tiffany himself. No expense has been spared."

The gift of Transportation is part of the fanlight. The description is "The gift of Transportation is envisioned as a cornucopia disgorging a steam locomotive. And that blue bit of glass might be the Montgolfier balloon."

The Spirit of Progress is part of the fanlight. The description is "It is part of her character to have bare shoulders like that."

The machinist is a bored man. He is in the Garage. He is wearing a grimy pair of overalls. He carries a wrench and a screwdriver. The current occupation of the machinist is "[The machinist] is making some adjustments to [the random thing which is part of the Victorian Car] with his [random thing carried by the machinist]"

The Victorian Car is a device in the Garage. A cast-iron steering wheel, a leather bucket seat, a horn, and a combustion engine are part of the Victorian Car. The seat is an enterable supporter.

```
Rule for writing a paragraph about a device (called X):
    let the subsequent mention be "Name";
    if the X is unmentioned:
        say "[The X] is here. ";
        let the subsequent mention be "It";
    if something is part of X:
        if the subsequent mention is "Name", say "[The X]
";
        otherwise say "[subsequent mention] ";
        say "[if a mentioned thing is part of X]also [end
if]features[if a mentioned thing is part of X], in addition to
[the list of mentioned things which are part of X],[end if]
[a list of unmentioned things which are part of X]";
        say ".".
```

```
Rule for printing the name of the steering wheel while
writing a paragraph about a person:
    say "steering wheel".
```

A supporter has some text called position. The position of a supporter is usually "None".

The Office is west of the Garage. The Office contains a desk. The desk has the position "A [desk] with several

dozen drawers stands in the center of the room". On the desk are some papers.

After printing the name of a supporter (called X) which supports an unmentioned thing:
    now X is unmentioned.

Rule for writing a paragraph about a supporter (called X):
    let the subsequent mention be "Name";
    if the position of X is not "None":
        say "[position of X]. ";
        let the subsequent mention be "It";
    if a mentioned thing is on X:
        say "Besides [the list of mentioned things which are on X], ";
        let the subsequent mention be "it";
    if the subsequent mention is "Name", say "[The X] ";
    otherwise say "[subsequent mention] ";
    say "holds [a list of unmentioned things which are on X]."

Section 2 - Schedule

The time of day is 4:38 PM.

At 4:42 PM:
    move the machinist to the Office;
    say "The machinist wanders into the Office to get some paperwork.";
    now every thing carried by the machinist is on the desk;
    now the current occupation of the machinist is "[The machinist] rifles through [the papers] on [the desk]".

At 4:43 PM:

move the young lady to the Garage;
if the young lady can be seen by the player,
say "An attractive young lady walks in from the street, and glances around as though she has never been here before."

At 4:45 PM:
if the young lady can be seen by the player,
say "With a not-quite-convincing air of innocence, [the young lady] happens to lean upon [the horn], which bleats loudly.";
otherwise say "There is a honk from the Garage[if the machinist can be seen by the player]. The machinist looks up with a frown[end if].";
now the horn is mentioned.

At 4:46 PM:
move the machinist to the Garage;
say "The machinist strolls from the Office into the Garage to find out what is going on.";
now the current occupation of the machinist is "[The machinist] is chatting with [the young lady]. He seems to be demonstrating the various features of [the car], including [the random thing which is part of the car]";
now the current occupation of the young lady is "[The young lady] is asking [the machinist] a number of questions about [the car]".

At 4:49 PM:
if the young lady can be seen by the player, say "[The machinist] gives [the young lady] his arm to climb into [the seat].";
move the young lady to the seat;
now the young lady is rapt;
now the current occupation of the young lady is "[The young lady] is turning [the steering wheel] from side to side";
now the current occupation of the machinist is "[The machinist] is leaning on the door of [the car], pointing out features to [the young lady]";
move the besotted expression to the machinist;
now the machinist is wearing the besotted expression.

At 4:52 PM:
    now the sober grey gown is unbuttoned at the neck;
    if the young lady can be seen by the player, say "[The young lady] murmurs something about the wilting heat, and undoes a button or two of her gown. The machinist's expression is comical, or would be, if you weren't annoyed."

Every turn when the player is in the Garage and young lady is on the seat:
    say "You are beginning to feel a little unnecessary in this scene."

Every turn when the player is in the Office and the young lady is on the seat:
    say "There's no sound at all from the other room, not even conversation."

Before going to the Garage when the young lady is on the seat:
    now the sober grey gown is tellingly dishevelled;
    move the young lady to the Garage;
    now the young lady is blushing;
    say "There is a flurry of movement as you enter the room.";
    now the current occupation of the young lady is "[The young lady] stands near the door, tapping her foot nervously";
    now the besotted expression is nowhere;
    now the current occupation of the machinist is "[The machinist] is leaning against [the car], looking smug".

Section 3 - Initially Out of Play

The besotted expression is a wearable thing. The description is "It looks foolish, doesn't it?"

The young lady is a bemused woman. She is wearing a sober grey gown and a pair of black boots. The current occupation of the young lady is "[The young lady] is running a gloved finger along the chassis of [the victorian car]"

Before printing the name of the young lady while writing a paragraph about a person:
    say "[mood of the young lady] "

The description of the grey gown is "Something about the perfect row of tiny buttons has the wrong effect -- at any rate, it is natural to wonder how long they take to undo." The gown can be buttoned almost to the chin, unbuttoned at the neck, or tellingly dishevelled.

Rule for printing the name of the gown when writing a paragraph about a person:
    say "sober grey gown ([sober grey gown condition])"

Test me with "z / look / look / z / look / west / east / z / look / z / look / z / look / west / east".

Example **Age of Steam**

The railway-station examples so far put together into a short game called "Age of Steam".

The following source is very short and simple, yet it already feels surprisingly interesting in play, because something is going on which the player does not control but must observe. The single scene both starts and finishes.

"Age of Steam"

The Station is a room. "Eynforme Halt is a raised platform fringed with cowslip: a whistle-stop with no more than a signal and a water-tank."

The Flying Scotsman is fixed in place. "The Flying Scotsman, fastest train in the world, is now at a dead standstill."

Train Stop is a scene. Train Stop begins when the player is in the Station for the third turn. Train Stop ends when the time since Train Stop began is 3 minutes.

When Train Stop begins:
    now the Flying Scotsman is in the Station;
    say "The Flying Scotsman pulls up at the platform, to a billow of steam and hammering."

When Train Stop ends:
    now the Flying Scotsman is nowhere;
    say "The Flying Scotsman inches away, with a squeal of released brakes, gathering speed invincibly until it disappears around the hill. All is abruptly still once more."

Instead of entering the Flying Scotsman, say "Alas, the [time when Train Stop began] arrival is only to take on water, not to set down or pick up."

Test me with "z / z / z / enter flying scotsman / z / z".

ExampleMeteoric I and II

A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

The game below begins at half past eleven, and one turn later, it's meteor time:

```
"Meteoric I"


The time of day is 11:30 PM.


At 11:31 PM:
    now the meteor is in the great outdoors;
    say "A meteor streaks across the sky.".


The great outdoors is a region. The Spanish Balcony is
east of the Inner Court. The Court and Balcony are in
the great outdoors. Inside from the Court is the
Swimming Pool.


The meteor is a backdrop. Instead of doing something
to the meteor, say "The meteor is no longer visible, now
nothing more than a memory."


Test me with "wait / wait / examine meteor / west /
examine meteor / in / examine meteor".
```

Or for something a little slower-moving and with no after-image:

```
"Meteoric II"
```

```
The time of day is 4:30 PM.


At 4:31 PM:
    now Phobos is in the great outdoors;
    say "Phobos rises from the western horizon."
At 10:06 PM:
    now Phobos is nowhere;
    say "Phobos sets over the eastern horizon."


The great outdoors is a region. The Martian Balcony is
east of the Inner Court. The Court and Balcony are in
the great outdoors. Inside from the Court is the Heavy
Water Swimming Pool.


Phobos is a backdrop. Instead of doing something to
Phobos, say "Phobos orbits a mere 6000km above you,
which is practically touching range for astronomy. On
the other hand, astronomy isn't all that practical."


Test me with "wait / wait / examine phobos / west /
examine phobos / in / examine phobos".
```

Though we should not really use Earthly time-keeping, since the Martian day is about half an hour longer than ours.

Example **Space Patrol - Stranded on Jupiter!** *WI*

We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

American radio adventure series of the 1950s were unobtrusively sponsored by breakfast cereals, as the following modest example demonstrates. Note that the scene-changing for the commercial break needs to know nothing about the actual programme it breaks into: if Part I were replaced with a different Space Patrol episode, Part II need not be

changed at all.

"Space Patrol #57 - 1953-10-31 - Stranded on Jupiter!"

Use scoring.

Part I - Serial

Red Spot is a room. "You are in the middle of a vast red oval plain. Overhead, the thick Jovian clouds swirl menacingly, and a fine acrid dust falls instead of rain." Some acrid dust is scenery in the Red Spot. The description of the dust is "The rust-colored dust coats every surface. You've no idea how deep it goes."

Instead of going in Red Spot, say "As you once told Cadet Lucky, Jupiter's a mighty big planet, maybe bigger than Iowa. Why, the Red Spot alone stretches out almost to the horizon."

The player wears a silver uniform and rubber boots. The player carries a shovel and an Analscope. The description of the Analscope is "As you recall from Space Patrol #9 - 1952-11-29 - The Electronic Burglar, the Analscope is a device for locating buried metals. That's what guided you all the way from the orbit of Uranus. (Oh, all right, Neptune.) If only you hadn't crashed!"

The metal plate is a fixed in place container. It is openable and closed. In the metal plate is some water. The description of the metal plate is "Stamped with the distinctive logo of the previous mission."

Instead of examining the player, say "Your hair clumps together stickily, thanks to the dustfall."

Digging is an action applying to one thing. Understand "dig [something]" or "dig in [something]" as digging.

Instead of digging the dust, try looking under the dust. Instead of looking under the dust when the metal plate is not visible: move the metal plate to the location; say "You brush aside the dust underfoot and -- what were the odds? -- it turns out that you landed just where the previous landing party did, thirteen ill-fated years ago. Here is the metal plate that covers their original well.

But wait! Called by the clanging of your shovel on the plate, a band of Jovian pterodactyls swoop down to attack! You're totally defenceless! You don't have a hope! You're absolutely finished!"; increase the score by 10; move K-Klak to Red Spot.

K-Klak the Pterodactyl is an animal. "K-Klak, leader of the Jupiter Pterodactyls, menaces you. A terrifying creature of scaly wings, with a dragon's tail, K-Klak stands... about 1/8th of an inch tall." Instead of doing something to K-Klak, say "K-Klak makes a frankly panicky noise and leaps backwards, out of your way."

After opening the metal plate: increase the score by 10; say "You have found water! You're saved! K-Klak makes a (very cautiously) pleased noise. Now to find the stolen Brainograph, and track down the crook with the thick Jewish accent and his henchmen with their thick Polish accents..."; end the story finally.

The maximum score is 20.

Part II - Cereal

When play begins, say "Instant Ralstons and Regular Ralstons, the hot whole-wheat cereals in the red and white checkerboard packages present... SPACE PATROL... High adventure in the wild vast reaches of space... Missions of daring in the name of interplanetary justice... Travel into the future as Buzz Corey, Commander-in-Chief of the..."

Last score is a number that varies. Every turn: now the last score is the score.

Ralstons Ad is a scene. Ralstons Ad begins when score is not the last score. Ralstons Ad ends when the Ricechex is consumed.

Include Basic Screen Effects by Emily Short.

When Ralstons Ad begins:
    center "*** We'll be back in just a moment! ***";
    pause the game;
    strip the player;
    move the player to the Kitchen.

When Ralstons Ad ends:
    center "*** And now, back to today's exciting adventure ***";
    pause the game;
    restore the player.

Saved location is a room that varies. Locker is a container. Wardrobe is a container.

To strip the player:
    now every thing carried by the player is in the locker;
    now every thing worn by the player is in the wardrobe;
    now saved location is location.

To restore the player:
    now every thing carried by the player is in the Kitchen;
    now every thing in the locker is carried by the player;
    now every thing in the wardrobe is worn by the player;
    move the player to saved location.

The Space Patrol Kitchen is a room. "The nerve center of the Space Patrol! This is where cadets fill up with their SUPER-FUEL. North leads to the astro control room, while back south is the cargo hold." A breakfast bowl is in the Kitchen. In the bowl is Ricechex. Ricechex is edible. The Ricechex can be consumed or uneaten. The Ricechex is uneaten.

Instead of going north in Kitchen: say "[refusal to leave]". Instead of going south in Kitchen: say "[refusal to leave]".

Instead of examining the player when Ralstons Ad is happening: say "You are currently being played by a generically attractive person of about 30, with very good teeth and well-kept nails."

After eating the Ricechex: say "That's right folks, always start your day the SPACE PATROL way with a tasty

```
bowl of Ricechex, Wheatchex or good hot Ralstons.
Mmmm Mmmm. You just can't get enough of the sugary
goodness in Ricechex, Wheatchex and good hot
Ralstons."; now the ricechex is consumed.



Instead of tasting the Ricechex:
    say "Wow! *wolf-whistle* Man oh man oh man!
Yumm-y!"



To say refusal to leave:
    repeat through Table of Refusals:
        say "[nope entry][paragraph break]";
        blank out the whole row;
        rule succeeds;
    say "You can't. Eat your Ricechex."



Table of Refusals
nope
"You can't go that way in the limited universe of this
sponsored message."
"Or that way."
"You've already tried that!"
"Why would you want to walk away when you have an
alluring bowl of Ricechex right here?"



Test me with "n / i / x me / x dust / dig dust".



Test ad with "n / s / n / s / n / i / x me / get bowl / taste
ricechex / eat ricechex".



Test ending with "x plate / x k-klak / open plate".
```

Episode 57 of "Space Patrol" was actually called "Iron Eaters Of Planet X", just in case the reader feels that any of the foregoing unfairly traduces a work of thoughtful science fiction.

Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

For some games, it makes sense to organize the entire game around scenes rather than around locations, moving the player when a new scene begins and laying out new props.

To this end, we might extend Inform's default handling of scenes so that each scene has properties to indicate prop lists and locations, and move objects in and out of play automatically as the scenes change. For instance:

```
"Entrevaux"

Part 1 - Procedure

A scene has a room called the starting location.

A scene has a list of objects called the scenery props.

A scene has a list of objects called the inventory props.
```

The starting location is the room to which the player should be moved; scenery props are things that need to be put there when the scene begins; inventory props, things that are given to the player when the scene begins; and the description some printed text to introduce the new scene. We may still occasionally need to have recourse to special "When the Dancing-Lesson begins..." rules for individual scenes, but for the most part this allows us to set scenes up in a consistent and predictable way.

Another point that might be slightly less obvious: sometimes we want to announce a change of location to the player when the scene starts, and sometimes we don't. In particular any scene that starts "when play

begins" should probably not explicitly describe the entered room, since that would duplicate the description automatically produced on the first turn of play. So we add a property to track whether any given scene should be announcing its location:

```
A scene can be location-silent or location-loud.
```

And let's say that we also want to announce each new scene as another "chapter" of the game in play, with a pause before the scene begins.

Here we include "Basic Screen Effects" because it will allow us to pause the game for a keypress, then clear the screen before each new chapter:

```
Include Basic Screen Effects by Emily Short.


The chapter counter is a number that varies.


First when a scene (called the current scene) which is
not the Entire Game begins:
    if chapter counter is greater than 0:
        pause the game;
    increment chapter counter;
    say "[bold type]Chapter [chapter counter]: [current
scene][roman type]";


Last when a scene (called the current scene) which is
not the Entire Game begins:
    repeat with item running through the scenery props of
the current scene:
        move the item to the starting location of the current
scene;
    repeat with item running through the inventory props
of the current scene:
        move the item to the player;
    if the location is not the starting location of the current
scene:
        if the current scene is location-loud:
            move the player to the starting location of the
current scene;
```

otherwise:
move the player to the starting location of the current scene, without printing a room description.

At the end of each scene, we strike the set and remove all the loose objects from play.

When a scene (called the current scene) ends:
repeat with item running through things which are not fixed in place:
if the item is not the player:
now the item is nowhere.

Part 2 - Scenario

Entrevaux Station is a room. "The station building consists of a waiting room and a ticket-selling office so small that only one person can buy a ticket at a time. On the outside wall is a clock that runs twelve minutes late; but since the trains also run twelve minutes, give or take, behind their published schedule, this clock is helpful in establishing reasonable expectations. [paragraph break]Painted on the door is the logo of the Chemin de Fer de Provence, the only railway in France that is not part of the SNCF."

The Hillside Tower is a room. "It's very dark in here, lacking artificial lighting, but from the rough rectangular window you can see a slice of hillside and a little of the river Var."

The window is scenery in the Hillside Tower. The description is "Through it can be seen a slice of wooded hillside and exposed grey-brown cliff. You are in the southern French foothills of the Alps, and the territory is dry. The only respite is the river Var, a milky blue at this time of year, running shallowly over mud and large stones far below your window." Understand "view" or "slice of hillside" or "hillside" or "hill" or "river" or "var" or

"mud" or "stones" or "large stones" as the window.

A used ticket is a thing. The description is "A piece of receipt paper indicating that you have paid the one-way fare of 9 euros from Nice. There is a hole punched through it."

A one-euro coin is a thing. The description is "It's a bimetal coin, brassy around the rim and silvery in the center. One side shows western Europe, with unusual prominence given to the UK, and the other side Leonardo da Vinci's four-armed, four-legged man having a nice stretch. It's dated 2002."

Some re-enactors are a person. "Milling about one end of the station is a crowd of medieval re-enactors." The description is "They're dressed in a somewhat aimless range of styles roughly honoring the period of 900-1500 AD. One gentleman is wearing a knobby leather cap; which is a good thing, because there is a rooster standing on his head." Understand "men" or "man" or "gentleman" or "rooster" or "reenactors" or "crowd" or "medieval" or "woman" or "women" as the re-enactors.

A kidnapper is a person. "Your kidnapper is watching you from the corner with his arms folded. You have the impression he's just marking time until someone more important arrives." The description is "He does not look at all like the kidnapping sort, but more like a sommelier at a superior restaurant: he wears a black pinstriped suit and has nicely-manicured hands."

The trolley is an enterable fixed in place container. "The 'train' on which you arrived is really just a single car, more like a trolley than a proper train." Understand "car" or "train" as the trolley. The description is "It has a glass front, so you can see ahead while riding: an innovation among trains."

Arrival is a location-silent scene. "After many days[']
journey, you have arrived at last in Entrevaux, a walled
medieval town now chiefly of interest to tourists and
crusade re-enactors."
    The starting location of Arrival is the Entrevaux
Station.
    The scenery props of Arrival are { re-enactors, trolley
}.
    The inventory props of Arrival are { the used ticket,
one-euro coin }.

Arrival begins when play begins. Arrival ends when the
time since Arrival began is 2 minutes.

Abduction is a location-loud scene. "You check into the
Hotel Vauban and sleep deeply enough; it was a long
and sticky trip to get here.

Then in the middle of the night something confusing
happens. You have the impression of strangers in your
room, and then a searing pain, and you don't come back
to yourself until midmorning of the following day..."
    The starting location of Abduction is the Hillside
Tower.
    The scenery props of Abduction are { kidnapper }.

Abduction begins when Arrival ends.

Test me with "i / x re-enactors / z / z / i / x him".

to the princess, as a break from the marriage-proposal scene.

"Pine"

Part 1 - The Set-up

This is mostly a repeat of what we have already seen, but for the sake of producing a playable scenario, we include it. The new material appears at Part 2.

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking [attitude][end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."; now Beauty is passive.

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but how?"

Instead of attacking an asleep person:
now the noun is awake;
say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
now the noun is awake;
say "[The noun] slowly stirs to wakefulness!"

Instead of throwing water at an asleep person:
now the second noun is awake;
now the noun is nowhere;
say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour [something] on [something]" or "splash [something] at/on [something]" as throwing it at.

Table of Conversation

topic	reply
"Sleep/dream/nightmare/nightmares/sleep"	"Sleep, dream, nightmare, nightmares. Do you ask me anything else?"

'Ghastly nightmares,' she remarks. You nod politely.

"'Not really,' she replies, edging away from you. So much for that angle."

"marriage/love/wedding/boyfriend/beau/lover"

'Sorry,' you say. 'This is a bit weird since we've just met, but, um. Would you like to get married?'

She looks at you nervously. 'Do I have to?'

"'I, er,' she says. 'I hope I'm not supposed to marry you or something.'"

"marriage/love/wedding/boyfriend/beau/lover"

'I was told I was going to marry you and inherit the kingdom,' you say, apologetically. 'Would that be very bad?' This could be awkward, considering your family circumstances -- you did promise your mother that everything would be better, after this --

'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.

"You try to think how to point out that it's been a hundred years since she last saw her boyfriend..."

'Do you think I could go look for him? I'm seeing him, you see, and I think I've been... sick... for a while, so he might be worried.'

You try to think how to point out that it's been a hundred years since she last saw her boyfriend. And try not to think how awkward things would be in your family if she refuses to marry you."

"marriage/love/wedding/boyfriend/beau/lover"

'You've been up here for a hundred years,' you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

"She shakes her head mutely."

She goes to the window and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

```
You shrug, uncomfortable."


Instead of asking an awake beauty about a topic listed
in the Table of Conversation:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.


Instead of telling an awake beauty about something: try
asking the noun about it.


Instead of asking an asleep person about something:
    say "[The noun] snores."


Marriage Proposal ends when the number of filled rows
in the Table of Conversation is 0.


Every turn during Marriage Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.


Every turn: now Beauty is active.


When Marriage Proposal ends: end the story saying
"This is going to take some explaining."
```

So far we haven't much of a chance to affect matters and make them better. Suppose we'd like to add an element to the conversation where

we're allowed to tell Beauty about past events -- and explore them a bit; and if the first retelling doesn't go quite as planned, we're allowed to revisit these scenes to hit them with a bit more emphasis.

Part 2 - Flashbacks

```
Instead of asking an awake beauty about a topic listed
in the Table of Flashback Material:
    now Beauty is passive;
    say "[reply entry][paragraph break]".
```

```
A fact is a kind of thing. The family circumstances is a
fact. A fact can be known or unknown. A fact can be
current or past.
```

Once known, a fact remains known permanently -- this could be useful if we wanted to make some rules about how Beauty acts when she knows different information. By contrast, a fact is only "current" if it is the last thing discussed. Since a player can mention a fact over and over, he can make it "current" again and again, and thus reactivate the flashback.

```
Table of Flashback Material

topic                reply
"[if family circumstances is unknown]I wish you'd give me
[otherwise]I don't think you fully understand the
[family circumstances],' you say, your jaw tensing. 'I
wouldn't ask if it weren't for my [family
circumstances]...'[otherwise]I don't think you fully understand the
[family circumstances],' you say.[end if]"
```

```
After printing the name of a fact (called target): now the
target is current; now the target is known.
```

This "After printing the name..." rule will be explained later in the chapter on activities; for now, it is enough to know that whenever family circumstances is mentioned in the table of flashback material, this rule will automatically be called. Now the terms under which the flashback

happens:

> Poverty flashback is a recurring scene. Poverty flashback begins when family circumstances is current. When poverty flashback begins: strip the player; move the player to the Woodcutter's Shack.

Note the "recurring" here: we want the player to be able to revisit this scene as needed.

> The Woodcutter's Shack is a room. "Your family lives in a shack in the forest. There are holes in the roof, and in the winter the snow comes in -- rain, too, for that matter. The walls aren't very well-boarded, and don't keep out the wind, and even though you live in the middle of dense woods, you can never gather enough fuel to keep this place fully heated. And then there's the stench. Pigs wander freely in and out, and your three youngest brothers play on the floor."

> Pigs are an animal in the shack. The pigs are scenery. The description is "They really are very grubby, dirty animals, but what's worse than that, the value of pigs has declined a lot over the last few decades. This is hard to explain to someone who has been out of touch with the world for a while, but keeping pigs for meat is a dubious prospect when there's less and less for them to forage on." Instead of smelling the pigs: say "They smell the way animals do, when they live among their own refuse."; increase the pity of Beauty by 2.

> The brothers are a man. The description of brothers is "Hans, Franz, and Lukas. Twins and then the baby. So young, and growing up fatherless; and soon to be orphaned entirely, if your mother's health does not improve." Understand "brother" or "twin" or "twins" or "baby" or "franz" or "hans" or "lukas" as the brothers.

> The untidy bed is scenery in the Shack. Mother is a

woman on the untidy bed. The description of mother is "She is wasting away of a slow disease, her skin stretched tautly over bone. She hasn't been the same since your father left." On the bed is a folded letter.

The description of the letter is "Many times read over and creased, the letter explains how your father has gone away with a wealthy countess and will not return. Your mother was not able to read it herself, of course, and had to have it explained to her by the parish priest. Now she keeps it by the bed and crumples it in her fits of delirium."

Instead of kissing or touching Mother for the first time:
say "You place a gentle kiss on her feverish brow. She looks up at you, her oldest -- yes, never mind that bit -- with a look of sincere trust and admiration.

'You'll find a way through this for us,' she says, squeezing your fingers. 'I know you will.'"; increase the pity of Beauty by 3.

Instead of kissing Mother: say "You have no more heart-rending memories of affection to recount; that one incident will have to serve, for rhetorical purposes."

Instead of waiting in the Shack: say "The wind blows sharply through the walls."

Instead of attacking someone in the Shack:
say "Though sometimes the conditions of your life make you grouchy and impatient, you would never dream of striking a member of your own family. But from time to time you do feel the temptation."

```
Beauty has a number called pity. After examining
something in the Woodcutter's Shack, increment the pity
of Beauty. After examining mother, increase the pity of
Beauty by 2. After examining the letter, increase the pity
of Beauty by 3.


Poverty flashback ends when waiting or the time since
poverty flashback began is five minutes.


When Poverty flashback ends:
    now family circumstances is past;
    say "...you finish describing the miserable
circumstances of your home life, and allow your
attention to return to the present.";
    restore the player;
    now Beauty is passive;
    if Beauty is sympathetic, say "'Oh dear!' she says.
'What a dreadful life!' She wrings her hands. 'No wonder
you are eager to improve your lot...! But --' Her brow
clears, a new thought occurring. 'You needn't marry me,
you know! We could arrange it differently! I am certain
that my father would give you a large reward, instead,
and then I would not be separated from my current
boyfriend!'";
    otherwise say "She makes a disgusted face, but she
doesn't seem nearly so heart-wrung as you had hoped
to make her. Tough audience, these modern
princesses."


Definition: Beauty is sympathetic if the pity of Beauty is
greater than 4.


To say attitude:
    if Beauty is sympathetic, say "distressed on your
behalf";
    otherwise say "a little confused".
```

And the following is the same as in the Space Patrol example as well: we need a systematic way to remove the player's possessions, then put everything back when the flashback is over:

```
Saved location is a room that varies. Locker is a
container. Wardrobe is a container.


To strip the player:
    now every thing carried by the player is in the locker;
   now every thing worn by the player is in the
wardrobe;
   now saved location is location.


To restore the player:
   now every thing carried by the player is in the
location;
   now every thing in the locker is carried by the player;
   now every thing in the wardrobe is worn by the
player;
   move the player to saved location.


Test me with "x beauty / wake beauty / pour water on
beauty / ask beauty about sleep / tell beauty about
poverty / smell pigs / x mother / x letter / kiss mother /
ask beauty about marriage / z / z".
```

Because we haven't changed the endings of the "Marriage Proposal" scene, there is still only one way for this scenario to work out; but at least now the player has the opportunity to alter Beauty's attitude a bit (or not) before the game is done.

Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the Princess has understood the point.

Suppose in addition to our pathetic little family history, we have another secret to convey to the Princess, this one a little more peculiar. She either gets it or she doesn't; once she gets it, we do not revisit that flashback, though it is still possible to keep visiting the poverty flashback.

"Pine"

Part 1 - The Set-up

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside you, looking [attitude][end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."; now Beauty is passive.

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but how?"

Instead of waiting in the presence of an asleep person (called snorer): say "You are alone with the sound of [the snorer] snoring sonorously."

Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"

Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour [something] on [something]" or "splash [something] at/on [something]" as throwing it at.

Table of Conversation

"Sleep/dream/nightmare/sleep"	reply

'Oh really, I'm sorry, I didn't realize. You must be very much in love.'

'Actually, she ran away from me. So I'm not so much for that angle."

"Sorry, I have to go. This is a little weird because we just met, but, um. Would you like to get married?'

She looks at you nervously. 'Do I have to? I mean, I'd rather not.'

"Well, this escalated quickly. I'm supposed to marry you or something.' Uh oh."

"I was told we were getting married and I don't want the kingdom,' you say, apologetically. 'Would that be very bad?' This could be awkward, considering your family circumstances -- you did promise your mother that everything would be better, after this --

'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.

"Do you think you could point out that it's been a hundred years since you last saw her boyfriend." for a while, so he might be worried.'

You try to think how to point out that it's been a hundred years since she last saw her boyfriend. And try not to think how awkward things would be in your family if she refuses to marry you."

"Marriage? How would you feel about that if he's no longer interested in you?' you suggest.

She gives you the look of a wounded squirrel. 'My father might not approve of my love for the kitchen boy, but his heart is faithful and true!' she exclaims.

"Right, suppose you that he's still a, she says rather whisperw'n't know fade, consider, reopening your fingers. Maybe such letter as my William, who works in the kitchen, after all she might be more fair about it than if she understood telling you this in there in her base-born romances."

Right then. Perhaps you'd better tell her your secret, in exchange?"

"Your lover, perhaps? A friend, maybe?" you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

She shakes her head and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

You shrug, uncomfortable."

Instead of asking an awake beauty about a topic listed in the Table of Conversation when Marriage Proposal is happening:

now Beauty is passive;
say "[reply entry][paragraph break]";
blank out the whole row.

Instead of telling an awake beauty about something: try asking the noun about it.

```
Instead of asking an asleep person about something:
    say "[The noun] snores."


Marriage Proposal ends in failure when the number of
filled rows in the Table of Conversation is 0.


Every turn during Marriage Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.


When Marriage Proposal ends in failure: end the story
saying "This is going to take some explaining."


Part 2 - Flashbacks


Instead of asking an awake beauty about a topic listed
in the Table of Flashback Material:
    now Beauty is passive;
    say "[reply entry][paragraph break]".


A fact is a kind of thing. A fact can be known or
unknown. A fact can be current or past.


The family circumstances is a fact. The secret identity is
a fact. The printed name of secret identity is "secret".
```

Table of Flashback Material

reply
"[if family circumstances is unknown]I wish you'd give me a little more information about your family circumstances,' you say, your jaw tensing. 'I wouldn't ask if it weren't for my [family circumstances]...'[otherwise]I don't think you fully understand the [family circumstances],' you say.[end if]"
"[if dramatic revelation ended in failure]'Look,' you say, trying again. 'I'm not [secret identity].'[otherwise]You clear your throat and allow your voice to stray upward, into its natural register and out of this husky false tenor you've been affecting. 'There's, er, something you should know about my [secret identity],' you say...[end if][if dramatic revelation ended in success] She looks impatient. 'I get it, you know,' she says. 'I'm not stupid.'"

After printing the name of a fact (called target): now the target is current; now the target is known.

Poverty flashback is a recurring scene. Poverty flashback begins when family circumstances is current. When poverty flashback begins: strip the player; move the player to the Woodcutter's Shack.

The Woodcutter's Shack is a room. "Your family lives in a shack in the forest. There are holes in the roof, and in the winter the snow comes in -- rain, too, for that matter. The walls aren't very well-boarded, and don't keep out the wind, and even though you live in the middle of dense woods, you can never gather enough fuel to keep this place fully heated. And then there's the stench. Pigs wander freely in and out, and your three youngest brothers play on the floor."

Pigs are an animal in the shack. The pigs are scenery. The description is "They really are very grubby, dirty animals, but what's worse than that, the value of pigs has declined a lot over the last few decades. This is hard to explain to someone who has been out of touch

with the world for a while, but keeping pigs for meat is a dubious prospect when there's less and less for them to forage on." Instead of smelling the pigs: say "They smell the way animals do, when they live among their own refuse."; increase the pity of Beauty by 2.

The brothers are a man in the shack. The brothers are scenery. The description of brothers is "Hans, Franz, and Lukas. Twins and then the baby. So young, and growing up fatherless; and soon to be orphaned entirely, if your mother's health does not improve." Understand "brother" or "twin" or "twins" or "baby" or "franz" or "hans" or "lukas" as the brothers.

The untidy bed is scenery in the Shack. Mother is a woman on the untidy bed. The description of mother is "She is wasting away of a slow disease, her skin stretched tautly over bone. She hasn't been the same since your father left." On the bed is a folded letter.

The description of the letter is "Many times read over and creased, the letter explains how your father has gone away with a wealthy countess and will not return. Your mother was not able to read it herself, of course, and had to have it explained to her by the parish priest. Now she keeps it by the bed and crumples it in her fits of delirium."

Instead of kissing or touching Mother for the first time:
    say "You place a gentle kiss on her feverish brow. She looks up at you, her oldest -- yes, never mind that bit -- with a look of sincere trust and admiration.

'You'll find a way through this for us,' she says, squeezing your fingers. 'I know you will.'"; increase the pity of Beauty by 3.

Instead of kissing Mother: say "You have no more heart-rending memories of affection to recount; that one incident will have to serve, for rhetorical purposes."

Instead of waiting in the Shack: say "The wind blows sharply through the walls."

Instead of attacking someone in the Shack:
    say "Though sometimes the conditions of your life make you grouchy and impatient, you would never dream of striking a member of your own family. But from time to time you do feel the temptation."

Beauty has a number called pity. After examining something in the Woodcutter's Shack, increment the pity of Beauty. After examining mother, increase the pity of Beauty by 2. After examining the letter, increase the pity of Beauty by 3.

Poverty flashback ends when waiting or the time since poverty flashback began is five minutes.

When Poverty flashback ends:
    now family circumstances is past;
    say "...you finish describing the miserable circumstances of your home life, and allow your attention to return to the present.";
    restore the player;
    now Beauty is passive;
    if Beauty is clever and Beauty is sympathetic:
        say "'I understand,' she says slowly. 'Yes, I do. I'll do it.' She takes a deep breath and looks at you. 'We will be king together! and your family will be royalty!'";
        end the story finally;
    otherwise:
        if Beauty is sympathetic, say "'Oh dear!' she says. 'What a dreadful life!' She wrings her hands. 'No wonder you are eager to improve your lot...! But --' Her brow clears, a new thought occurring. 'You needn't marry me,

you know! We could arrange it differently! I am certain that my father would give you a large reward, instead, and then I would not be separated from my current boyfriend!'";
otherwise say "She makes a disgusted face, but she doesn't seem nearly so heart-wrung as you had hoped to make her. Tough audience, these modern princesses."

Definition: Beauty is sympathetic if the pity of Beauty is greater than 4.

To say attitude:
if Beauty is sympathetic, say "distressed on your behalf";
otherwise say "a little confused".

Saved location is a room that varies. Locker is a container. Wardrobe is a container.

To strip the player:
now every thing carried by the player is in the locker;
now every thing worn by the player is in the wardrobe;
now saved location is location.

To restore the player:
now every thing carried by the player is in the location;
now every thing in the locker is carried by the player;
now every thing in the wardrobe is worn by the player;
move the player to saved location.

Part 3 - The Other Secret

This time, we're waiting for the princess either to understand or not understand -- so we don't want to rerun the scene once it has happened successfully.

```
Beauty has a number called clue count.


Dramatic revelation is a recurring scene. Dramatic
revelation begins when attempting confidence.


To decide whether attempting confidence:
    if dramatic revelation ended in success, no;
    if secret identity is current, yes;
    no.


When dramatic revelation begins:
    strip the player;
    say "You reminisce about one of the many stops on
the way here: you had a long journey from your
homeland, and it wasn't made any easier by your
poverty, the inability to afford decent inns or plentiful
food or any kind of ride along the way.";
    move the player to the Forest Clearing;
    move the pack to the player; now the player wears
the trousers; now the player wears the shirt.


Forest Clearing is a room. "It's mid-autumn in your
memory, the pool clear and cold, gold and red leaves
floating on the surface."


The pool is scenery in the Clearing. Understand
"reflection" or "surface" or "water" as the pool. "The pool
is cold but beautiful, and the stopping place a welcome
rest." The leaves are scenery in the clearing. The
description is "Bright gold and orange and red: it's been
a sharply chilly autumn, as you have reason to know in
detail."
```

The trousers and the shirt are wearable things. The pack is a container. The pack contains ale, food, and skirt. A distraction is a kind of thing. The ale, the food, the pair of trousers, and the shirt are distractions. The description of a distraction is usually "[The item described] is not the point of this story." The shirt and the trousers are wearable. The description of the trousers is "Borrowed from your oldest brother, who is only a year younger than you. They are too long for your legs and overly snug at the hip, but no one around here pays much attention to fashion, and you're getting away with it, more or less." After examining the trousers, increment the clue count of Beauty.

Instead of examining the player during dramatic revelation:
    increment the clue count of Beauty;
    say "You cannot see yourself without reflection, but you can feel your hair loose and unbound over your shoulders."

Rule for printing the name of the skirt while taking inventory: say "one skirt you have not been able to bring yourself to part with". The description of the skirt is "Made for you by your mother, and it looks quite pretty on you. If your primary plan does not work, you may be forced to wear it again, and hope to catch a male eye... but with luck that will not be necessary." After taking inventory: increment clue count of Beauty.

Swimming is an action applying to nothing. Understand "swim" or "dive" as swimming.

Instead of swimming in the presence of the pool:
    increment clue count of Beauty;
    say "You consider going for a swim, but don't dare be caught unclad and unarmed, not here. There are too many men around, and any of them discovering you here would surely take advantage."

Instead of searching or drinking the pool:
    increment clue count of Beauty;
    say "You lean over the pool and look carefully at your reflection, your hair loose and unbound, falling around your face in waves. (That should surely give it away!)"

Instead of waiting during dramatic revelation: say "You wait for the penny to drop, for her to understand."

Dramatic revelation ends in failure when waiting or the time since dramatic revelation began is five minutes. When dramatic revelation ends in failure:
    now secret identity is past;
    restore the player;
    now Beauty is passive;
    say "She wrinkles her nose. 'I don't understand!' she says. 'What are you trying to tell me?'

    You could weep for womankind. But you don't quite dare spell it out in so many words, not when someone might come up the stair and overhear a chance revelation."

Dramatic revelation ends in success when Beauty is clever. When dramatic revelation ends in success:
    restore the player;
    now Beauty is passive;
    say "'You're -- a girl? Like me?'

'Not much like you,' you say, glancing over her petite frame and pert nose. 'But female, at any rate.'"

Definition: Beauty is clever if the clue count of Beauty is greater than 2.

And now, since we don't really want to return to the rest of the 'marriage proposal' scene once she has learned our ID:

```
Marriage proposal ends in distraction when Dramatic
Revelation ends in success.
```

```
Compromise proposal is a scene. Compromise proposal
begins when Dramatic Revelation ends in success.
When Compromise Proposal begins: now Beauty is
passive.
```

```
Instead of asking an awake beauty about a topic listed
in the Table of Secondary Conversation when
Compromise Proposal is happening:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.
```

```
Every turn during Compromise Proposal:
    if Beauty is active and Beauty is visible:
        repeat through Table of Secondary Conversation:
            say "[quip entry][paragraph break]";
            if the number of filled rows in the Table of
Secondary Conversation is greater than 1, blank out the
whole row;
            make no decision.
```

```
Every turn: now Beauty is active.
```

Notice that we moved the re-activation rule down here so that the Compromise Proposal rule would fire first. There are other more complicated ways of handling order of every turn rules than by relying on text sequence alone; but we will save that for a later chapter. For now it is sufficient to depend on the order in which the rules are declared.

```
Table of Secondary Conversation
```

teply

"Wiessyoag/more sctf eldpbedeo'uth'/iderterency'ce,' you go on. 'You could carry on whatever romances you wished, without your father noticing.' (Probably. You'll let the pragmatic details of this work themselves out later, and hope that any children she has will look vaguely like you.)"

'Wrong,' you say, feeling a little annoyed. 'But you see why marrying me wouldn't be an interference. You could carry on whatever romances you wished, without your father even noticing.'"

"Ifngo/u'rean'thinking that a woman can't be the prince -- and then king -- well, there was a woman Pope, once.'

"She lockssawe in her forehead does not go away. 'But if everyone thinks you are a man... later you would be king!'

Before she can go on, you say, 'There was a woman Pope, once. Compared to that, a woman king is nothing.'"

"decision/proposal/marriage/choice?"

"Her pretty nose twitches again, which you are coming to recognize as a sign of hard mental labor. 'I think I see,' she says. [final decision]"

To say final decision:
if Beauty is sympathetic:
say "She considers the matter silently for some minutes, then says: 'I will do it. My beloved William will be so glad!' You imagine that William's feelings on the

matter will be a tad more complex than that, but do not bother quashing her exuberance...";
end the story finally;
otherwise:
say "'I still don't quite understand why you would want this so badly as to go to all that trouble,' she admits uneasily. Evidently you have not explained enough to her about the poverty of your home life."

Test me with "x beauty / pour water on beauty / ask beauty about sleep / tell beauty about poverty / smell pigs / x mother / x letter / kiss mother / ask beauty about marriage / tell beauty about identity / x me / look in water / i / z / ask beauty about marriage".

ExampleFate Steps In

Fate entity which attempts to make things happen, by hook or by crook, including taking preliminary actions to set the player up a bit.

One of the nice things about before rules for actions is that they allow us to express some planning for characters other than the player: we've already seen how this works, a bit. But we could also use before rules to write plans for an abstract story-driving entity, rather than for other individual characters. This story-driver could be in charge of all the non-player characters, as well as spontaneous or natural changes in the environment, shaping the narrative around the player's behavior.

The following example is a very simple one, but the same concept could be worked out in a great deal more complexity, with all sorts of alternative procedures available to our story-manager:

"Fate Steps In"

Fate is a woman. After deciding the scope of the player: place Fate in scope. The description of Fate is "Not smiling." Instead of doing something other than examining to fate: say "As if."

Every turn: try fate tripping.

Tripping is an action applying to nothing.

Carry out someone tripping:
    if something dangerous (called the trap) is in the location:
        say "Lise chooses this moment to lick her fingers -- it's not gross, it's natural, you decide -- stand up, and head for the door. Unfortunately, her path crosses directly over [the trap]. There is a vaudevillesque moment where you try to warn or catch her; the next moment she's on the floor, looking shocked and also in quite a lot of pain. 'I'm not sure,' she says to you steadily but with unfocused gaze, 'but I think I might have broken my tailbone.'";
        end the story saying "Well, she's paying attention to you now".

Before someone tripping when the location does not contain a dangerous thing: try the person asked making a mess instead.

Making a mess is an action applying to nothing.

Carry out someone making a mess:
    let calamitous object be a random visible supporter which supports at least three things;
    if calamitous object is a supporter:
        say "[The calamitous object] tips over, spilling [the list of things on calamitous object] all over the place.";
        move the calamitous object to the location;
        now every thing on the calamitous object is in the location.

Definition: a thing is dangerous if it is not the carton and it is not the table and it is not a person.

Before someone making a mess when a safe supporter (called target) is visible:
    if Lise carries something, try Lise putting a random thing carried by Lise on the target instead.

Instead of someone making a mess when the tray is on the table:
    say "Just at that moment, a large blond man-thing in a red jacket walks more or less through you, and you come into violent contact with the table, knocking [the list of the things on the table] onto the floor.";
    now every thing on the table is in the location;
    now every thing on the tray is in the location instead.

Definition: a supporter is safe if the number of things on it is less than two.

McQuerry Dining Hall is a room.

The table is scenery in the dining hall. The table is a supporter.

Lise Fitzwallace is a woman in the Dining Hall. "Lise is at the nearest table, not apparently paying any attention to you." The description of Lise is "A capella singer, women's rugby champion, general object of attention from all genders. Unlikely to notice you unless fate smiles broadly." Lise carries a fork, a napkin, an empty glass, and a plate of half-eaten eggplant parmesan.

Report Lise putting something on something:

say "Lise, still deep in thought, absently puts [the noun] on [the second noun]." instead.

The carrying capacity of the player is 2. The carton of chocolate milk is in the Hall. "There's a carton of milk beside you, which you set down for a moment -- but you do want it."

Instead of taking something when the player carries the tray:
say "You've got both hands full with this tray."

The player carries the tray. On the tray is some macaroni and some overdone chicken. The macaroni and the chicken are edible. The tray is portable.

Test me with "get milk / put tray on table / get milk".

Test again with "drop tray".

Chapter 5: The Viewpoint Character

Contents of *The Inform Recipe Book*
Chapter 4: Time and Plot
Chapter 6: Commands
Indexes of the examples

§5.1. The Human Body

By default, Inform gives the player character (and every other person) a simple unitary body, one without hands or feet or any other defined parts. In many games this is adequate; but in others it is not enough, and we may want to endow all people with some more specific physical features, as in

```
A face is a kind of thing. A face is part of every person.
```

Once we've done this, we may invite ambiguities if the player types LOOK AT FACE; it is this challenge that is addressed in **The Night Before**.

rBGH gives the player a random height and then uses this to determine how the room should be described around him.

Slouching lets the player (and other characters as well) take different sitting, standing, and lying down positions.

Start of Chapter 5: The Viewpoint Character
Back to Chapter 4: Time and Plot: §4.6. Plot Management
Onward to §5.2. Traits Determined By the Player

Example 254: ★ **rBGH** The player character's height is selected randomly at the start of

play.

Example 396: ★★ **Slouching** A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

Example 59: ★★★ **The Night Before** Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

---

§5.2. Traits Determined By the Player

Some IF tries to make the viewpoint character more congenial to the player by allowing some customization.

Identity Theft demonstrates asking the player to supply the viewpoint character's name.

Pink or Blue demonstrates a way to let the player choose a gender at the start of play: this will mostly be interesting if the rest of the story makes some use of the player's choice. Since that example is written expressly to demonstrate included Inform 6 code, however, we may find it more congenial to generalize from the more flexible **Baritone, Bass**.

This is not the only way to go - as we'll see in the next section, there's also something to be said for making the viewpoint character a strongly distinct creature with well-defined preferences and attitudes.

Start of Chapter 5: The Viewpoint Character

Back to §5.1. The Human Body

Onward to §5.3. Characterization

Example 416: ★ **Identity Theft** Allowing the player to enter a

name to be used for the player character during the game.

Example 309: ★★★ **Baritone, Bass** Letting the player pick a gender (or perhaps other characteristics) before starting play.

Example 451: ★★★ **Pink or Blue** Asking the player to select a gender to begin play.

---

§5.3. Characterization

Much of the personality of the player character in IF emerges from what he can and cannot (or will and will not) do; part of the pleasure of playing a character arises from this opportunity for role-playing and role-exploration. Some characters are consciousless daredevils, willing to jump off cliffs, crawl through narrow gaps, and rob widows if the player commands it; others are repressed neurotics who barely dare to speak to other characters or touch anything that doesn't belong to them.

Finishing School and **Dearth and the Maiden** both treat the case of a character constrained by good manners and a sense of polite society: the former forbids only one action, while the latter condemns a whole range of them.

Constraining the character is only the half of it: we might also want to think about what sorts of unusual actions that character might be especially likely to take, and account for these. Of course, major actions that affect the story world will require some thought and implementation work, and we should consider carefully before making the player a character like, say, the Noble of Glamour, a spirit in human form who can charm all comers, transform bespectacled secretaries into divas, and cause spontaneous cloudbursts of scarlet glitter.

But even simple humans have some characteristic traits and gestures. We will probably want to write some characteristic reaction to EXAMINE ME, as demonstrated in **Bad Hair Day**. We might provide a few pieces of clothing or props that aren't strictly critical in the story, like a policeman's helmet or a feather boa:

```
The player is wearing a policeman's helmet.
```

We can liven up the interactive aspect of characterization if we give the player a little scope for role-playing: this may mean responding to gestures, like

> Understand "bite nails" as a mistake ("Your only nail remaining is the one on your left thumb, and you're saving it for the AP Calculus exam.").

(Of course, we would need to have hinted to the player that nail-biting is characteristic of his character.)

See **Clothing** *for more on dressing characters up*

See **Saying Complicated Things** *for conversation, another area in which the player character's personality might come into play*

Start of Chapter 5: The Viewpoint Character
Back to §5.2. Traits Determined By the Player
Onward to §5.4. Background

Example 73: **Finishing School** The "another" adjective for rules such as "in the presence of another person".

Example 85: **Bad Hair Day** Change the player's appearance in response to EXAMINE ME.

Example 109: **Dearth and the Maiden** Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and seemliness.

---

§5.4. Background

In IF, as in all interactive storytelling, an essential problem is that the player does not begin the story knowing everything that the player character should, and so may implausibly bumble through situations that the player character should be quite comfortable in. If the player character has friends, an unusual job, a home or environment we're not familiar with, a secret past, these will all be a blank to the player.

Some games get around this by making the player character an amnesiac, or positioning him as a newcomer to a strange world in which his disorientation is explicable; but there are stories that cannot be told this way, and so we need other methods of getting the player to know what the player character already does.

Our first opportunity to inform the player about the player character is in the opening text of a story:

```
When play begins:
    say "The funeral is exactly a month ago now, but Elise's shoes are
still on the shoe tree."
```

We may also want to write descriptions of objects to give extra background information the first time the player encounters them:

```
A thing can be examined or unexamined. A thing is usually
unexamined. After examining something: now the noun is examined;
continue the action.
```

```
The description of the newspaper is "A rolled-up newspaper[if
unexamined], and thus a symbol of your newly-single state: Elise
always had it open and the Local Metro section next to your plate by
the time you got out of the shower[end if]."
```

To expand on this, we could give the player a THINK ABOUT or REMEMBER command, with which he can call up information about people he meets or references he encounters in descriptions, so that he could (for instance) next type REMEMBER ELISE. **Merlin** demonstrates one way to implement a character with memory; **One of Those Mornings** puts a twist on this by letting the player FIND things which he knows his character possessed at some time before the story started.

Start of Chapter 5: The Viewpoint Character
Back to §5.3. Characterization
Onward to §5.5. Memory and Knowledge

Example 277: ★ **Merlin** A REMEMBER command which accepts any text and looks up a

response in a table of recollections.

Example 301: **One of Those Mornings** A FIND command that allows the player to find a lost object anywhere

---

§5.5. Memory and Knowledge

All of us carry around in our heads an (incomplete, imperfect) model of the world around us: an idea of where we left the keys, whether the oven is on or off, how many clean pairs of socks are left in the drawer, what we look like in our best pair of jeans. The differences between that mental model and reality are to some degree a reflection of personal character: our forgetfulness, our wishful thinking, our innocence or cynicism.

By default, Inform does not keep track of the player character's knowledge (or any other character's knowledge, for that matter) as a separate thing from the model world, relying on descriptive prose rather than modeling to introduce these quirks of characterization.

All the same, there are often times when we would like to keep track of discrepancies between the world model and the narrator's mental model. Perhaps the most common way to do this is simply to mark everything that the player encounters as "seen" when the player first examines it, thus:

```
A thing can be seen or unseen.

Carry out examining a thing:
    now the noun is seen.
```

or -- to have things remembered from the first moment they're mentioned in a room description:

```
Rule for printing the name of something (called the target):
    now the target is seen.
```

The mental model need not always be accurate, of course. We might, for instance, have occasion to keep track of where the player character last saw something, even if

the object has since been moved; or keep track of falsehoods the player character has been told in conversation; or make the player refer to a character as "the bearded man" until he is properly introduced.

Included with Inform is the extension Epistemology, by Eric Eve, which provides one way of tracking this kind of information. Epistemology distinguishes between items that the player character has seen, because they're objects in a room the player has been to, and items that are familiar to the player for other reasons, such as a quest item he knows about but hasn't found yet, or an abstract conversation topic. Anything that is either seen or familiar is counted as "known".

Modeling what the player does and does not know is only half the job, of course: we also need that information to affect the behavior of the story in plausible ways.

One obvious occasion to use player character knowledge is in the output of descriptions. We might want to respond to actions differently depending on what the player has previously done, as in **Tense Boxing**, or change the way we describe objects in light of new knowledge about them, as in **Zero**. **Casino Banale** takes that idea much further, with a whole system of facts that can be narrated to the player in a somewhat flexible but interdependent order, as the player looks at relevant objects or notices them in room descriptions.

Along similar lines, we may want an object to change its name for the player depending on what the player knows. That name change should affect both what Inform displays and what it understands about the object. For instance:

```
An Amherz Amulet is a thing. It can be known or unknown. It is
privately-named.

The printed name is "[if known]Amherz Amulet[otherwise]lizard-
shaped pewter charm[end if]".

The description is "[if known]It's a unique and magically powerful
pewter charm shaped like a lizard[otherwise]It's some cheap tacky
pewter charm shaped like a lizard. At least, as far as you can tell --
it's pretty grubby[end if]."

Understand "amherz" or "amulet" as the Amulet when the Amulet is
known.
```

```
Understand "lizard" or "lizard-shaped" or "pewter" or "charm" as the
Amulet when the Amulet is unknown.


Instead of rubbing the amulet when the amulet is unknown:
    say "You rub off a bit of the dirt, and... what do you know? It's
actually the priceless and fabulously powerful Amherz Amulet!";
    now the Amherz Amulet is known.
```

Finally, the player's knowledge may affect how the story interprets commands, in the determining what is called "scope". When Inform tries to make sense of something the player has typed, it makes a list of everything that the player is allowed to refer to at the moment, and then checks whether all of the objects in the player's command refer to items in that list. Only things that are "in scope" are open for discussion.

If the player mentions an object that is not "in scope" -- say, a red hat left behind in the next room -- Inform will issue the response "You can't see any such thing." This is also Inform's reply if the player mentions a nonsense object ("EXAMINE FURSZWIGGLE") or an object that does not exist in the story world at all ("EXAMINE CELL PHONE" in a story set in Carolingian France).

This is not the only possible way for interactive fiction to handle such communication. Some games will respond differently to EXAMINE RED HAT and EXAMINE FURSZWIGGLE, saying in the first case something like "You can't see that now" and in the second "I don't know the word 'furszwiggle'."

The drawback of such behavior is that the player can make premature discoveries. If he hasn't found a sword yet, but thinks there may be a sword later in the story, he can type EXAMINE SWORD and see from the response whether his guess is correct. Nonetheless, there are people who prefer this alternative exactly because it does expose the limits of the story's understanding, preventing fruitless attempts to use a word that is not recognized at all. If it is desirable, there is an extension that will reproduce this behavior in Inform as well.

Using Inform's default behavior, however, scope is an ad-hoc way of keeping a list of things that are common knowledge between the story and the player. The player knows many things that the story might not (like what a cell phone is); the story knows a few things the player may not (like the fact that there is a sword in an as-yet unvisited room). Neither of those things can fruitfully enter into commands because they have no mutually agreed-upon referent.

By default, Inform assumes that "scope" includes only those things that are currently visible by line of sight. This works pretty well for a wide range of situations, but there are still plenty of occasions when we want to admit that the story and the player share a knowledge of things not seen. GO TO THE KITCHEN might be a useful command even when the player can't currently view the kitchen. ASK FRED ABOUT THE

FOOTPRINTS should perhaps work even when the footprints are far away in the garden. SMELL STINKY CHEESE might need to work even when the cheese is invisibly locked away in a porous container but is exuding a stench. In a dark room, the player can't see his own inventory, but he should still remember that he's carrying it and be able to mention it. And sometimes we might want the story to acknowledge that the player is referring to an object that he has seen somewhere, even if that thing is now out of sight.

In practice, we have two ways to tinker with scope: we can change the scope for a specific command, using a token with any, as in

```
Understand "go to [any room]" as approaching.
Understand "find [any thing]" as finding.
Understand "ask [someone] about [any known thing]" as interrogating
it about.
```

Or we can add areas and items to scope for all commands, as in

```
After deciding the scope of the player when the surveillance camera
is switched on:
    place the jail cell in scope.
```

Puncak Jaya demonstrates understanding references to characters who are currently off-stage.

See **Helping and Hinting** *for objects tagged with a "seen" property when the player first encounters them*

See **Getting Acquainted** *for a character whose name is changed during the course of play as the player gets to know him better*

See **Room Descriptions** *for more ways to change the description of a room depending on player experience*

See **Going, Pushing Things in Directions** *for ways to understand the names of distant rooms and move towards them*

See **Character Knowledge and Reasoning** *for models of knowledge for other characters than the player*

See **Sounds** *for ways of tracking audible objects separately from visible ones*

See **Lighting** *for ways to change what the player knows about and can*

manipulate in dark rooms

See **Clocks and Scientific Instruments** *for a telescope that lets the player view objects in another location*

See **Continuous Spaces and The Outdoors** *for more on seeing into adjacent locations*

Start of Chapter 5: The Viewpoint Character
Back to §5.4. Background
Onward to §5.6. Viewpoint

Example 149: ★ **Tense Boxing** An overview of all the variations of past and present tenses, and how they might be used.

Example 320: ★ **Puncak Jaya** When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

Example 148: ★★★ **Zero** A box which called "horribly heavy box" after the player has tried to take it the first time.

Example 360: ★★★ **Casino Banale** Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

---

§5.6. Viewpoint

Inform automatically creates a character for the player - a bland, personality-free

entity at the outset, as we've seen. But there is no reason why the player need stick to this same identity throughout the story. Conventional fiction often jumps from one viewpoint character to another, and so can IF.

To do this at the most elementary level, we simply at some point

```
now the player is Janine;
```

where Janine is a person we've already defined in the code. Now the player is in whatever location Janine inhabits, carries whatever Janine carries, and wears whatever Janine is wearing. **Terror of the Sierra Madre** shows off this effect, and also demonstrates how to make the command prompt remind the player which character he currently controls. Some games instead give this information in the status line or after the name of the location when looking, producing output like

```
The Bottomless Acherousia (as Charon)
```

We could do the same by adding a line such as

```
After printing the name of a room while constructing the status line or
looking:
    say "[roman type] (as [the player])"
```

Of course, we'll need a good deal of other work to make Janine a distinct person from whichever character the player was before. The distinction may come from changed capabilities of the new character, which we can express through new rules about actions; e.g.,

```
Instead of listening when the player is Janine:
    say "Your childhood accident left you unable to hear any but the
loudest noises. Currently there is only silence."
```

Janine may also have new, different perspective on her surroundings, expressed through the descriptions of the things she looks at; **Uncommon Ground** makes a "by viewpoint" token for text alternatives, allowing us to tag our descriptions to indicate which variations should be shown to which viewpoint characters. **The Crane's Leg 1** and **2** offer more elaborate and specialized ways of customizing the player character's observations to depend on how he relates (physically and in attitude) to the things around him.

If we want to change the tense and person of narration from the conventional present

second person, we may do this as well:

```
When play begins:
    now the story viewpoint is first person plural;
    now the story tense is past tense.
```

Though this only changes the form of the text produced automatically by Inform (responses such as "you can't go that way" might become, say, "I couldn't go that way"), and all author-written text in the story must be written in the tense and person intended.

Start of Chapter 5: The Viewpoint Character

Back to §5.5. Memory and Knowledge

Onward to Chapter 6: Commands: §6.1. Designing New Commands

Example 390: **The Crane's Leg 2** A description text generated based on the propensities of the player-character, following different rulebooks for different characters.

Example 455: **Uncommon Ground** Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

Example 52: **The Crane's Leg 1** A description text that automatically highlights the ways in which the object differs from a standard member of

its kind.

Example 121: ★★★ **Terror of the Sierra Madre** Multiple player characters who take turns controlling the action.

Examples from Chapter 5: The Viewpoint Character

Start of this chapter
Chapter 6: Commands
Indexes of the examples

★ **ExamplerBGH**
The player character's height is selected randomly at the start of play.

WI

As with ordinary numbers, we can choose random units when this is useful:

"rBGH"

The Pharmaceutical Testing Facility is a room. "A [if the player is short]large [end if][if the player is tall]cramped [end if]white space with sterile counters and a[if the player is tall]n uncomfortable little[end if] stool. There is also a mirror, behind which someone must be watching you. But you can't see through to that."

A counter, a one-way mirror, and a stool are scenery in the Facility. The stool is an enterable supporter. The counter supports a plate.

Height is a kind of value. 5 feet 11 inches specifies a height. 5'11 specifies a height. A person has a height.

Definition: a person is tall if its height is 6 feet 0 inches or more.

Definition: a person is short if its height is 5 feet 4 inches or less.

When play begins:
    now the height of the player is a random height between 5 feet 2 inches and 6 feet 4 inches;
    now the right hand status line is "[height of player]".

Instead of examining the player:
    say "You, Test Subject, are [height of the player] tall."

The growth pill is a kind of thing. A growth pill is always edible. The description is usually "It is leaf-green and has a reassuring logo of a curling vine on the side. Nothing to worry about, nothing at all." Two growth pills are on the plate.

After eating the growth pill:
    increase the height of the player by 0 feet 6 inches;
    say "Your spine does something frightening and painful, and you find yourself looking down on the room from a wholly new angle.";
    try looking.

Test me with "examine me / eat pill / examine me / eat pill / examine me".

A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

Suppose we want to let the player explicitly sit, stand, or lie down on different enterable objects. (Normally Inform treats all these actions as entering, but there may be cases where we want to distinguish between the player sitting on a chair and the player standing on it.)

Our implementation gives each kind of enterable a range of allowed postures, and one preferred posture. If the player enters the supporter or container without specifying a posture (as in ENTER CHAIR), he will be put in the preferred posture. If he explicitly says, e.g., STAND ON CHAIR, he will be put in the standing position as long as standing is a posture that suits the chair.

```
"Slouching"

Section 1 - Posture Rules

A posture is a kind of value. The postures are seated,
standing, and reclining.

A person has a posture. The posture of a person is
usually standing.

A supporter has a posture. A container has a posture.

Posture-permission relates various things to various
postures. The verb to allow means the posture-
permission relation.
```

Understand the commands "stand" and "sit" and "lie" as something new.

Understand "sit on/in [something]" as sitting on.
Understand "lie on/in [something]" as lying on.
Understand "stand on/in [something]" as standing up on.

Sitting on is an action applying to one thing.
Lying on is an action applying to one thing.
Standing up on is an action applying to one thing.

Carry out an actor sitting on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not seated, try the actor taking position seated;
        otherwise follow the report taking position rules.

Carry out an actor lying on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not reclining, try the actor taking position reclining;
        otherwise follow the report taking position rules.

Carry out an actor standing up on:
    if the holder of the actor is not the noun, silently try the actor entering the noun;
    if the holder of the actor is the noun:
        if the actor is not standing, try the actor taking position standing;
        otherwise follow the report taking position rules.

Understand "lie down" as lying down.

```
Understand "sit down" or "sit" or "sit up" as sitting down.
Understand "stand" or "stand up" as standing up.


Lying down is an action applying to nothing.
Sitting down is an action applying to nothing.
Standing up is an action applying to nothing.


Taking position is an action applying to one posture.


Instead of an actor lying down:
    try the actor taking position reclining; rule succeeds.
Instead of an actor sitting down:
    try the actor taking position seated; rule succeeds.
Instead of an actor standing up:
    try the actor taking position standing; rule succeeds.


Check an actor taking position:
    if the holder of the actor is not a room and the holder
of the actor does not allow the posture understood:
        if the actor is the player:
            say "You can't take that position [in-on the
holder of the actor].";
        otherwise if the actor is visible:
            say "[The actor] can't take that position.";
        stop the action.


Check an actor taking position:
    if the posture understood is the posture of the actor:
        if the actor is the player:
            say "You are already [the posture understood].";
        otherwise:
            if the actor is visible, say "[The actor] is already
[the posture understood].";
        stop the action.


Carry out an actor taking position:
```

```
    now the posture of the actor is the posture
understood.


Report someone taking position (this is the position-
report rule):
    say "[The actor] is now [the posture of the actor][if the
holder of the actor is not the location of the actor] [in-on
the holder of the actor][end if]."


Report taking position:
    say "You are now [the posture of the player][if the
holder of the player is not the location] [in-on the holder
of the player][end if]."


To say in-on (item - a thing):
    if the item is a container, say "in [the item]";
    otherwise say "on [the item]".


Carry out an actor exiting (this is the departure-posture
rule):
    let N be the holder of the actor;
    if N is a container or N is a supporter,
        now the posture of the actor is the posture of N;
    otherwise now the posture of the actor is standing.


The departure-posture rule is listed after the standard
exiting rule in the carry out exiting rulebook.
The departure-posture rule is listed after the standard
getting off rule in the carry out getting off rulebook.


Carry out an actor entering something (this is the arrival-
posture rule):
    if the noun is a container or the noun is a supporter,
        now the posture of the actor is the posture of the
noun.
```

The arrival-posture rule is listed after the standard entering rule in the carry out entering rulebook.

```
Check an actor going somewhere:
    if the actor is in a room and the actor is not standing:
        say "([if the actor is not the player][the actor] [end
if]first standing up)[command clarification break]";
        silently try the actor taking position standing;
        if the actor is not standing, stop the action.
```

Section 2 - Some Generic Kinds

A chair is a kind of supporter. A chair is always enterable. Every chair allows seated and standing. A chair is usually seated.
A sofa is a kind of supporter. A sofa is always enterable. Every sofa allows seated, standing and reclining. A sofa is usually seated.
A hammock is a kind of container. A hammock is always enterable. Every hammock allows seated and reclining. A hammock is usually reclining.

Section 3 - The Scenario

The Resort is a room.

The banana hammock is a hammock in the Resort. The stone bench is a sofa in the resort.

Clark is a man in the Resort. A persuasion rule: persuasion succeeds.

```
Rule for writing a paragraph about someone (called
target):
    say "[The target] is [posture] [if the holder of the
target is the location]nearby[otherwise][in-on the holder
of the target][end if]."


Rule for writing a paragraph about something which
encloses an unmentioned person (called target):
    carry out the writing a paragraph about activity with
the target instead.


Test me with "sit on bench / stand on bench / get up / lie
on hammock / sit up / g / clark, sit on bench / look /
clark, lie down / g / look / clark, get up / look / clark, lie
down / look / enter bench".
```

ExampleThe Night Before

Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

Suppose that we're going to give every person in the game a nose, but we want references to a nose always to mean the nose of someone *else*, if the player is with one other person. Moreover, on some occasions we're going to be in sight of Rudolph, so actions directed at an unspecified nose should always prefer his.

This relies on a somewhat advanced technique from the Understanding chapter, but since it may become useful with assemblies and body parts, it is worth mentioning here.

```
"The Night Before"


The North Pole is a room. "Here it is: the famous Pole.
From here you can go south (or south-south, or south-
```

south-by-south); or, alternatively, take refuge inside a red-and-white-striped cabin." The cabin is scenery in the North Pole. Instead of entering the cabin, try going inside.

Santa is a man in the North Pole. "Santa is pacing around in the snow and trying to psych himself up for the big night."

Inside from North Pole is the Candy Cane Cabin. The description of the Cabin is "Striped red and white, but nothing can make this place seem warm and inviting since Mrs. Santa ran off with the Tooth Fairy."

The Ice Shelf is south of North Pole. "The ice here has been smoothed into a kind of runway for easy take-off, and ends in a cliff and cold arctic sea." Donner, Vixen, Blixen, and Rudolph are animals in the Ice Shelf.

A nose is a kind of thing. A nose is part of every person. The description of Santa's nose is "It's a bit ruddy. You don't like to mention it, but Santa's been dipping heavily into the Grey Goose since Mrs. Santa left town." The description of a nose is usually "Not terribly exciting." The description of Rudolph's nose is "See how it glows!"

Next, we'll teach Inform some vocabulary to distinguish between the player and everyone else:

Definition: a person is other if it is not the player.

Definition: a thing is selfish if it is part of the player and the player can see an other person.

```
Instead of examining a selfish nose:
    say "You cross your eyes, but can't get a good look."
```

Here is the part that actually determines the preferences. "Does the player mean..." can result in five outcomes: "it is very unlikely", "it is unlikely", "it is possible" (the neutral default), "it is likely", and "it is very likely". This is discussed in greater detail in the Understanding chapter. Here, we want to discourage references to the player's own nose and encourage references to the nose of Rudolph, so:

```
Does the player mean doing something when the noun
is a selfish nose or the second noun is a selfish nose: it
is very unlikely.

Does the player mean doing something to Rudolph's
nose: it is very likely.
```

And this part is just for decoration:

```
Rule for writing a paragraph about Rudolph:
    say "The reindeer are already harnessed and waiting
impatiently. The brilliance of [Rudolph]'s nose casts an
eerie red glow over [the list of unmentioned animals in
the location]."

Test me with "x nose / x my nose / x santa's nose / in / x
nose / out / s / x my nose / x nose / x rudolph's nose / x
donner's nose".
```

Example Identity Theft

Allowing the player to enter a name to be used for the player character during the game.

Let's say we want to allow the player to enter any name he likes for his character. Moreover, we want to reject very long names (which are likely to be mistakes anyway), and we want to extract the player's chosen first

name from the rest.

```
"Identity Theft"

The player's forename is a text that varies. The player's
full name is a text that varies.

When play begins:
    now the command prompt is "What is your name? >
".

To decide whether collecting names:
    if the command prompt is "What is your name? > ",
yes;
    no.

After reading a command when collecting names:
    if the number of words in the player's command is
greater than 5:
        say "[paragraph break]Who are you, a member of
the British royal family? No one has that many names.
Let's try this again.";
        reject the player's command;
    now the player's full name is the player's command;
    now the player's forename is word number 1 in the
player's command;
    now the command prompt is ">";
    say "Hi, [player's forename]![paragraph break]";
    say "[banner text]";
    move the player to the location;
    reject the player's command.
```

We also want to postpone the proper beginning of the game until we've gotten the name:

```
Instead of looking when collecting names: do nothing.
```

```
Rule for printing the banner text when collecting names:
do nothing.


Rule for constructing the status line when collecting
names: do nothing.


Your Bedroom is a room. The printed name of Your
Bedroom is "[player's forename]'s Bedroom".


The player carries a letter. The description of the letter
is "Dear [player's full name], [paragraph break]You have
won the Norwegian Daily Lottery! ...".
```

If we are compiling for Glulx, this is enough to capture not only the player's name but also the capitalization he uses.

If we are compiling for the Z-machine, the player's input will unfortunately be reduced to lower case before we can inspect it. If we would like by default to capitalize the first letter of each word of the name, we might substitute the following after reading a command rule:

```
After reading a command when collecting names:
    if the number of words in the player's command is
greater than 5:
        say "[paragraph break]Who are you, a member of
the British royal family? No one has that many names.
Let's try this again.";
        reject the player's command;
    now the player's full name is the substituted form of
"[the player's command in title case]";
    now the player's forename is word number 1 in the
player's full name;
    now the command prompt is ">";
    say "Hi, [player's forename]![paragraph break]";
    say "[banner text]";
    move the player to the location;
    reject the player's command.
```

Example **Baritone, Bass**

Letting the player pick a gender (or perhaps other characteristics) before starting play.

The "reading a command" activity is rather advanced; for the moment, what we need to understand is that we're intervening in commands at the start of play and insisting that the player's first instruction to the game consist of a choice of gender. After that point, the gender will be set and play will proceed as normal.

In order to do the parsing, we define gender as a kind of value, and give several alternate names to each gender.

```
"Baritone, Bass"

Getting Started is a room.

Gender is a kind of value. The genders are masculine,
feminine, and unknown. Understand "male" or "man" or
"M" as masculine. Understand "female" or "woman" or
"F" as feminine.

A person has a gender. The gender of the player is
unknown.

When play begins:
    now the command prompt is "Please choose a
gender for your character. >".

After reading a command when the gender of the player
is unknown:
    if the player's command includes "[gender]":
        now the gender of the player is the gender
```

```
understood;
        if the gender of the player is unknown:
            say "This story requires a selection of male or
female. [run paragraph on]";
            reject the player's command;
        if the gender of the player is masculine, now the
player is male;
        if the gender of the player is feminine, now the
player is female;
        say "[line break]Thank you. We now begin...";
        now the command prompt is ">";
        move the player to Sandy Beach;
        reject the player's command;
    otherwise:
        say "Sorry, we're not ready to go on yet. [run
paragraph on]";
        reject the player's command.



Sandy Beach is a room.



Instead of examining the player when the player is
female:
    say "Congratulations, you are a girl!"



Instead of examining the player when the player is male:
    say "Congratulations, you are a boy!"
```

If we had a whole series of things to ask the player about, we might define a whole series of kinds of value

```
The vocal ranges are soprano, mezzosoprano,
contralto...
```

and use a "construction stage" variable to keep track of the current stage of character-construction, as in

```
After reading a command when the current construction
stage is choosing a vocal range:
    ...
```

Suppose we would like to allow the player to choose a gender for the main character. We'd also like this to happen at the beginning of the game and outside the main parsing sequence. "When play begins" seems like a good place to put this.

```
"Pink or Blue"


When play begins:
    say "Should your character be male or female? >";
    if men win, now the player is male;
    otherwise now the player is female;
    say paragraph break.
```

Now a piece of Inform 6 code handles the unusual input. It's not necessary to understand this to use it, and the code should work for any question you'd like to ask the player. The first three words in quotation marks ('male', 'M', 'man'...) correspond to positive feedback; the later three words correspond to negative feedback. So "to decide whether men win" will be true if the player types one of the first three, and false if he types one of the last three.

```
To decide whether men win:
    (- Question('male','M//','man','female','F//','woman') -)


Include (-


[ Question pos1 pos2 pos3 neg1 neg2 neg3
first_word_typed;
    while (true) {
        VM_ReadKeyboard(buffer, parse);
        wn = 1; first_word_typed = NextWordStopped();
```

```
        if (first_word_typed == pos1 or pos2 or pos3) rtrue;
        if (first_word_typed == neg1 or neg2 or neg3)
rfalse;
        print "Please choose ", (address) pos1, " or ",
(address) neg1, ". > ";
    }
];
```

-)

Instead of examining the player when the player is female:
    say "Congratulations, you are a girl!"

Instead of examining the player when the player is male:
    say "Congratulations, you are a boy!"

The Room of Self-Knowledge is a room. "Mirrors cover every available wall-surface of this hexagonal chamber, allowing you to examine yourself from all angles."

Example Finishing School

The "another" adjective for rules such as "in the presence of another person".

It is often useful to write action rules which apply only when the player is observed by a third party. "In the presence of a person", however, will react even if only the player is in the room, because the player is, of course, a person as well.

A convenient way around this problem is to define an "other" adjective:

"Finishing School"

The apple is an edible thing carried by the player.

The Hall is a room. Miss Wicket is a woman in the Hall.

The Dormitory is south of the Hall.

Definition: a person is another if it is not the player.

Instead of eating something in the presence of another person:
    say "Your mannerly upbringing prevents you from eating without a fork or knife in front of someone."

Test me with "eat apple / south / eat apple".

If we did not have "another" here, Inform would interpret even the player as a possible schoolmarm, leading to such lines as "yourself stares at you coldly...". Clearly not quite the thing.

ExampleBad Hair Day
Change the player's appearance in response to EXAMINE ME.

"Bad Hair Day"

The Foyer is a room. "A mirror hangs over the table, tempting you to check your appearance before going in

with all the others."

Instead of examining the player:
    say "Oh, stop fussing. You look fine."

Test me with "examine me".

Example **Dearth and the Maiden**

Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and seemliness.

The following example, indebted to the late Georgette Heyer, is suggestive:

"Dearth and the Maiden"

The Chequers Inn is a room. "The room is panelled and ceilinged in oak, with blue curtains to the windows and blue cushions on the high-backed settle by the fire."

An oil painting is in the Inn. "An oil painting hangs upon one wall, a lascivious work from the Indies in which a very bendy, sloe-eyed courtesan - but no."

A man called Mr Carr is in the Inn. "Standing bashfully aside is one Mr Carr, who we have been led to understand is by profession a Highwayman (yet whose visage oddly recalls Lord John Carstares, disgraced eldest son of the Earl of Wyncham)."

Kissing Mr Carr is unmaidenly behaviour. Doing something to the painting is unmaidenly behaviour.

Instead of unmaidenly behaviour in the Inn, say "How unmaidenly! Why, one might just as wantonly strip a rose of its petals, letting each fragrant leaf flutter slowly to the ground."

Test me with "examine painting / take painting / kiss mr carr".

---

277 ExampleMerlin *WI*

A REMEMBER command which accepts any text and looks up a response in a table of recollections.

"Merlin"

Understand "remember [text]" as remembering.

Remembering is an action applying to one topic.

Carry out remembering:
    say "Nothing comes to mind.".

Instead of remembering a topic listed in the Table of Recollections:
    say "[response entry][paragraph break]".

Table of Recollections

Response

"You've been worse, but not often: it's falling so hard now that the tin rattles and the runoff, on the low side of the roof, would be a tenable source of hydroelectric power."

"It's not as though you have any sort of light bulb in here to turn on, even if you could power it."

"Light bulbs, like so much else, are a thing of your past. Or is it your future? Tricky, the way the world loops round on itself."

"Living backwards has its drawbacks. A tendency to confuse and annoy your friends, being one; the total inability to maintain a stable relationship; and a deep dissatisfaction with most of the bodily processes people enjoy, since they ultimately make you hungrier, colder, or-- no point dwelling on it, really."

"It's not exactly backwards, now is it? It's more like a series of forwardses stuck back to back. As though someone had taken each track of a CD and put them in the exactly wrong order. You miss that. The music on demand."

The Inadequate Shelter is a room. "A piece of corrugated tin, leaned on two sticks, and pathetically augmented with a tire (on one side) and a cardboard box (on the side towards the wind). And that's what you've got between you and the driving rain.

At the moment rain is all you can remember, in fact."

Test me with "remember rain / remember power / remember light bulb / remember future / remember backwards".

---

301. Example One of Those Mornings *WI*

A FIND command that allows the player to find a lost object anywhere

Suppose that, contrary to the usual rules of interactive fiction, we want to allow the player to discover the locations of things he hasn't actually seen

yet:

```
"One of Those Mornings"

Understand "find [any thing]" as finding.

Finding is an action applying to one visible thing.

Carry out finding:
    if the player is carrying the noun:
        say "You're holding [the noun]!";
    otherwise:
        say "You left [the noun] [if the noun is on a
supporter]on[otherwise]in[end if] [the holder of the
noun]."
```

The holder of the noun can be a room, a supporter, or a container: the phrase is not picky. We would want to be a little more careful if it were ever possible for an item to have been "removed from play" in our game, since then the holder could be nothing, and that would have odd results. In this particular example, though, that will not arise.

And that's it, as far as the find command goes. The rest is local color.

```
The Exhibition Room is a room. It contains a closed
locked lockable transparent openable container called
the display case. The display case contains a priceless
pearl. The display case is scenery. The description of
the Exhibition Room is "By far the finest thing in the
room is a priceless pearl in a glass display case. It
should of course be yours[if key is not visible], if only
you can remember where you hid the key[end if]."

The silver key unlocks the display case.
```

A jade vase, a teak chest, a bronze teakettle, and a child's burial casket are openable closed containers in the Exhibition Room.

After taking the pearl:
    say "The pearl rolls into your hand, gleaming in the oblique light; your fortune is made.";
    end the story finally.

If we want to have the key found in different places when the game is replayed:

When play begins:
    let the space be a random container which is not the display case;
    move the silver key to the space.

Every turn:
    say "Your watch ticks with maddening loudness."

The time of day is 1:02 AM.

At 1:08 AM: say "The security guard arrives to find you fumbling about with keys. Curses."; end the story.

Test me with "find pearl / find teakettle / get teakettle / find teakettle / find key".

ExampleTense Boxing

An overview of all the variations of past and present tenses, and how they might be used.

Here we have a box that prints out its current state and its history each time we open and close it:

```
"Tense Boxing"


The Temporal Prism is a room. "A room of angled
mirrors, in whose surfaces you can see what is now;
what just was; what has always been. A final mirror is
broken and its frame gapes blackly."


The mysterious box is in the Temporal Prism. It is an
openable closed container.


To assess the box:
    if the box was not open, say "The box was not open.";
    if the box was open, say "The box was open.";
    if the box had not been open, say "The box had not
been open.";
    if the box had been open, say "The box had been
open.";
    if the box is not open, say "The box is not open.";
    if the box is open, say "The box is open.";
    if the box has not been open, say "The box has not
been open.";
    if the box has been open, say "The box has been
open."


Before opening the mysterious box:
    say "You are about to open the box.";
    assess the box.


Before closing the mysterious box:
    say "You are about to close the box.";
    assess the box.
```

```
After opening the mysterious box:
    say "You now open the box.";
    assess the box.


After closing the mysterious box:
    say "You now close the box.";
    assess the box.
```

Note that "was..." and "was not..." and so on may describe conditions more complicated than simple properties: we could equally well ask "if the box has been in the sack", "if the box had been carried by the player", and so on.

The past ("if the box was...") and past perfect ("if the box had been...") are especially useful for cases where we want to report on an action after the state of the item has changed; so, for instance:

```
After taking the mysterious box:
    if the box had not been carried by the player, say
"You lift the mysterious box for the first time.";
    if the box had been carried by the player, say "You
again pick up the mysterious box."


Test me with "open box / close box / open box / take
box / drop box / take box".
```

This is in many respects similar to a rule beginning "After taking the mysterious box for the first time...", but it is superior in most circumstances, for two reasons.

First, it will respond correctly even if the player has somehow carried the box before without taking it explicitly: for instance, if another character gave him the box, if the box were moved into his inventory as a result of another action, or if the player carried the box at the start of play. Inform begins its reckoning of time when the game begins, so if the box is defined as being open at the outset, "if the box has been open" will always be true.

Second, "after taking... for the first time" fires only the first time the player attempts to take something. If the player tried to take the box, failed, and then tried again later, the "for the first time..." rule would not fire; our "if the box has not been carried..." rule would.

When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

By default, when something is not present, Inform does not allow a player to refer to it. But there are times when we might like to acknowledge that the thing mentioned in a command does exist somewhere in the game; it just happens not to be on hand right now.

One way to do this is to make an object that appears everywhere and responds to the name of its owner only when the owner itself is not in view.

```
"Puncak Jaya"
```

```
A ghost is a kind of person. A man-ghost is a kind of
ghost. A man-ghost is always male. A woman-ghost is a
kind of ghost. A woman-ghost is always female.
```

We make the ghost a person rather than some other kind of thing so that it will be able to respond to commands such as KISS BOB or (even trickier) BOB, JUMP: if Inform did not recognize the ghost as an animate creature, it would not accept such input.

```
Representation relates one ghost to one person. The
verb to represent means the representation relation.
```

```
One man-ghost represents every man. One woman-
ghost represents every woman.
```

This is, technically, an assembly -- except instead of saying that every device has a button part, or that there are three daffodils in every garden room, the assembly is based on a non-physical relation that we just designed.

Based on the "representation" relation, we now devise a conditional relation that applies only when the represented thing is not itself in view:

```
Indication relates a ghost (called X) to a person (called
Y) when X represents Y and Y is not visible.


Understand "[something related by indication]" as a
ghost.


When play begins:
    now every ghost is in the concept-repository.


Instead of doing something to a ghost:
    say "You seem to have left [a random person which is
represented by the noun] behind."


Instead of doing something when the second noun is a
ghost:
    say "You seem to have left [a random person which is
represented by the second noun] behind."


The concept-repository is an open unopenable
transparent container. It is part of the air. The air is a
backdrop. It is everywhere.


Base of Puncak Jaya is a room. Temple, Kippax, and
Huizenga are men in Base. Peak of Puncak Jaya is
above Base of Puncak Jaya.


Test me with "x kippax / up / x kippax / kiss kippax /
kippax, hello".
```

Further complications of this example might require that the player meet a character before being able to refer to him or her.

A box which called "horribly heavy box" after the player has tried to take it the first time.

The names of objects might want to change also:

```
"Zero"
```

```
Weight is a kind of value. The weights are light,
manageable, and horribly heavy. Everything has a
weight.
```

```
A thing is usually manageable.
```

This does require a rule for an activity (see the chapter on Activities), but it's a fairly straightforward one:

```
Before printing the name of a horribly heavy thing
(called weighty object):
    if we have taken the weighty object, say "[weight] ".
```

```
The Nearly Empty Living Room is a room. "Nearly
everything is out, all of Helen's possessions and most of
yours." A man called Mr Zero is in the Nearly Empty
Living Room. "Mr Zero, despite being heavily paid to
assist in this operation, is giving you a look that clearly
conveys his lack of interest in budging even one more
item." The description of Mr Zero is "Many muscles, no
hair."
```

```
The book box is a horribly heavy thing in the Living
Room. The clothing box is a manageable thing in the
Living Room. A broom, a dustpan, some packing tape,
```

and a discarded newspaper are light things in the Living Room.

After taking a horribly heavy thing:
    say "Taken. (Oof.)"

Test me with "get clothing / look / get book box / look / i / drop book box / look".

ExampleCasino Banale

Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

In a work of interactive fiction that involves many new discoveries, we might want to change the way we narrate room descriptions and describe objects as the player learns new information.

One approach to this is to create a model of the facts we want the player to find out, and attach some narrative text to each. When a fact becomes relevant to the story, that narrative text is shown to the player. So:

"Casino Banale"

Section 1 - Procedure

First we create the concept of facts, and the idea that facts can make some things more important than others.

A fact is a kind of thing. A fact can be known or unknown. A fact can be ready to learn or hidden. A fact has some text called the narration.

```
Definition: a thing is narratively significant if it conveys
an interesting fact.

Definition: a thing is narratively dull if it is not narratively
significant.

Conveyance relates various things to various facts. The
verb to convey means the conveyance relation.

Definition: a fact is interesting if it is unknown and it is
ready to learn.
```

Now, we also need a way to tell Inform to introduce certain new facts when the right previous ones have been introduced. We'll create a "following" relation, according to which a new fact can be told to the player when the player has already learned all the facts it follows. This way, we can simulate the effect of putting together several pieces of evidence to come to a conclusion:

```
Following relates various facts to various facts. The verb
to follow means the following relation.

To say (new fact - a fact):
    say "[narration of the new fact]";
    now the new fact is known;
    repeat with possible outcome running through facts
which follow the new fact:
        if every fact which is followed by possible outcome
is known:
            now the possible outcome is ready to learn.
```

Next we need a way for the game to introduce these new facts. Let's say we want them to come up when the player examines something appropriate, or sees it in the room:

```
After examining something which conveys an interesting
```

```
fact (called discovery):
    say "[discovery][paragraph break]".


After choosing notable locale objects:
    repeat through the Table of Locale Priorities:
        if the notable-object entry is narratively significant:
            set the locale priority of the notable-object entry
to 1.


For writing a paragraph about a narratively significant
thing (called item):
    now the item is mentioned;
    let chosen fact be a random interesting fact which is
conveyed by the item;
    say "[chosen fact][paragraph break]".
```

The "after choosing notable locale objects" line here handles things so that any interesting conclusions we want to draw are always given first, followed by the less interesting description.

And finally, we need to give the player a little evidence to piece together:

```
Section 2 - Scenario


The Casino is a room.


Frince is a man in the Casino. The description is "Frince
is a friend of yours -- if you reckon friendship on the
same terms that one reckons a cat as a pet. He spends
time with you when he wants to, but if your wishes or
convenience ever run counter to a whim of his, it's the
whim that wins. Always. [paragraph break]He's also
wearing a somewhat ludicrous shirt."


Frince wears a ludicrous shirt. The description of the
ludicrous shirt is "Fine white fabric with satiny white
pinstripes: it's that expensive, effeminate look that
```

Frince is so fond of, and which -- combined with his name -- gives people completely the wrong idea about him."

Tim is a man in the Casino. The description is "You don't know Tim well. Kind of wall-flowerish. The only thing that seems to excite him is craps."

Penny is a woman in the Casino. The description is "Loud. Brash. Hot, probably, if you can look past the loud and brash."

Rule for writing a paragraph about a narratively dull person:
    let is-are-n be "is";
    if the number of unmentioned narratively dull people is not 1:
        let is-are-n be "are";
    say "[A list of unmentioned narratively dull people] [is-are-n] [one of]watching the croupier[or]following the spin of the roulette[or]chattering[at random][one of] breathlessly[or] impatiently[or][at random]."

Penny-annoying is a fact.
    It is ready to learn.
    The narration is "[if looking]Penny grimaces at you--[end if]Penny is the same woman who stepped on your toe in the buffet line. The third time, she blurted, 'You have big shoes, don't you?'"
    Penny conveys penny-annoying.

lipstick-smudges is a fact.
    It is ready to learn.
    The narration is "There are a couple of smudges of coral-colored lipstick on the collar."
    The ludicrous shirt conveys lipstick-smudges.

penny-wears-coral is a fact.
    It follows penny-annoying.
    The narration is "[if looking]Penny catches your eye again. [end if]The bright coral lipstick was really not a wise choice."
    Penny conveys penny-wears-coral.

Affair-with-penny is a fact.
    It follows lipstick-smudges and penny-wears-coral.
    The narration is "You avoid [if examining Frince]his[otherwise]Frince's[end if] eye. You need some time to adjust to the image of him making out with Penny in a storage closet before you can talk to him without appalled giggling."
    Frince conveys affair-with-Penny.

Test me with "x penny / x frince / x shirt / look".

Example **The Crane's Leg 2**

A description text generated based on the propensities of the player-character, following different rulebooks for different characters.

Names of rules can be listed in tables. This is convenient if, for instance, we decide that we'd like to swap the rules we use for a specific purpose, as in this continuation of our earlier example of automated description:

"The Crane's Leg, Grown Longer"

Material is a kind of value. The materials are wood, glass, stone, cloth, paper, clay, and metal. A thing has a material.

Color is a kind of value. The colors are red, orange, yellow, green, blue, indigo, violet, black, brown, and

white. A thing has a color. A thing is usually white.

A height is a kind of value. 3 feet 11 inches specifies a height. A thing has a height. Definition: a thing is tall if its height is 6 feet 0 inches or more. Definition: a thing is short if its height is 2 feet 0 inches or less.

Imitation relates various things to one thing (called the ideal). The verb to imitate means the imitation relation.

A table is a kind of supporter. A table is usually wood. The height of a table is usually 3 feet 8 inches. The ordinary table is a table. Every table imitates the ordinary table.

A rock is a kind of thing. A rock is usually stone. The ordinary rock is a rock. The height of a rock is usually 0 feet 3 inches. Every rock imitates the ordinary rock.

The description of a thing is usually "[comparison with ideal][run paragraph on]".

To say comparison with ideal:
    say "You observe [the noun]:[paragraph break]";
    choose row with character of the player in Table of Descriptive Reporting;
    follow instructions entry.

This is the comparative observation rule:
    let the sample be the ideal of the noun;
    if the sample is not a thing:
        say "Nothing special, really.";
        rule succeeds;
    if the material of the noun is not the material of the

```
sample:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun], and made of
[material of the noun].";
            otherwise say "Unusually tall at [height of the
noun], and made of [material of the noun].";
        otherwise:
            say "Distinct mostly in being made of [material of
the noun].";
    otherwise:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun].";
            otherwise say "Unusually tall at [height of the
noun].";
        otherwise:
            say "In every respect [a sample]."
```

The Pleasure Garden is a room. "At the riverbank, a pleasing garden, having many curving paths and one straight."

The low table is a table in the Pleasure Garden. The height of the low table is 2 feet 3 inches. On the low table is a yellow metal rock called a gold nugget. A willow is in the Pleasure Garden. The height of the willow is 20 feet 2 inches.

Understand "possess [any person]" or "be [any person]" as possessing.

Possessing is an action applying to one thing. Carry out possessing: now the player is the noun; say "You swap bodies!"

The crane is a person in the Garden. The height of the

crane is 4 feet 0 inches.

Table of Descriptive Reporting

interactions	
yourself	comparative observation rule
bird	observation rule

This is the bird observation rule:
    if the noun is shorter than the player, say "Small, like a duck[if the color of the noun is not white]; and [color of the noun][end if].";
    otherwise say "Supremely tall[if the color of the noun is not white] and [color of the noun][end if]."

Test me with "examine table / examine nugget / examine willow / possess crane / examine table / examine nugget / examine willow".

Example **Uncommon Ground**

Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

A slightly more challenging case than the "by atmosphere" example is one in which we want to create text variations depending on the identity of our player character.

What we want to do is build a switch statement in I6, one that looks something like

```
switch(player)
{
    yourself: print "quaint";
    Lolita: print "thrilling";
```

```
    default: print "squalid";
}
```

out of I7 that looks like this:

```
say "[show to yourself]quaint[to Lolita]thrilling[to
everyone else]squalid[end show]".



"Uncommon Ground"



The Mud Village is a room. "You stand at the center of a
[show to yourself]quaint[to Lolita]thrilling[to everyone
else]squalid[end show] mud village."



Leforge is a man in the Mud Village. Lolita is a woman
in the Mud Village.



Instead of waiting:
    if the player is Lolita, now the player is Leforge;
    if the player is yourself, now the player is Lolita;
    say "You jump bodies. Whoops!"



To say show to (N - a person) -- beginning
say_seen_by:
    (-
        switch(player)
        {-open-brace}
        {N}:
    -).



To say to (N - a person) -- continuing say_seen_by:
    (-
        {N}:
    -).
```

```
To say to everyone else -- continuing say_seen_by:
    (-
        default:
    -)



To say end show -- ending say_seen_by:
    (-
        {-close-brace}
    -)



Test me with "look / z / look / z / look".
```

---

52 **Example** **The Crane's Leg 1** *WI*

A description text that automatically highlights the ways in which the object differs from a standard member of its kind.

> When you see a long object, you don't have to think that it is too long if being long is the property given by the nature. It is proved by the fact that a duckling, having short legs, will cry if you try to draw them out by force, and that a crane, having long legs, will protest you with tears if you try to cut them with a knife. -- Zhuangzi

Suppose we have an extremely detailed world model in which every object is characterized by many features -- in this example, material and height, though one could add more. Suppose further that we would like to generate descriptions of these things automatically for the most part, drawing the player's attention only to those aspects of the object that are particularly interesting.

```
"The Crane's Leg"
```

```
Material is a kind of value. The materials are wood,
glass, stone, cloth, paper, clay, and metal. A thing has a
material.
```

```
A height is a kind of value. 3 feet 11 inches specifies a
height. A thing has a height. Definition: a thing is tall if
its height is 6 feet 0 inches or more. Definition: a thing is
short if its height is 2 feet 0 inches or less.
```

So far, we have generally dealt with cases where the property of a thing can be a number (such as 3), a value (such as brightness), or a unit (like height, here). It is also possible for a thing to have a property which names another thing, as in "The mother of the baby trout is the large trout" -- where "mother" is a property, and its value, in the case of the baby trout, is large trout. We would define such a property with a line such as "A fish has a thing called the mother."

In practice, though, this is a bit confusing as syntax; moreover, Inform has a much more powerful construct for talking about the ways in which one object relates to another object. A full discussion of this will have to wait for the chapter on Relations. For now, it is enough to say that we can do this:

```
Imitation relates various things to one thing (called the
ideal). The verb to imitate means the imitation relation.
```

This will allow us to declare that some objects imitate other objects, like so:

```
A chair is a kind of supporter. A chair is always
enterable. A chair is usually wood. The height of a chair
is usually 4 feet 0 inches. The ordinary chair is a chair.
Every chair imitates the ordinary chair.
```

```
A table is a kind of supporter. A table is usually wood.
The height of a table is usually 3 feet 8 inches. The
ordinary table is a table. Every table imitates the
ordinary table.
```

```
A rock is a kind of thing. A rock is usually stone. The
ordinary rock is a rock. Every rock imitates the ordinary
rock. The height of a rock is usually 0 feet 3 inches.
```

```
A jug is a kind of container. A jug is usually clay. The
ordinary jug is a jug. Every jug imitates the ordinary jug.
The height of a jug is usually 0 feet 8 inches.
```

Now each of these types has one ideal representative which has the fundamental attributes of its kind: the ordinary chair is the most chairlike chair imaginable, the ordinary table is the epitome of tableness, and so on. We are also allowed to refer to "the ideal of the chair", thanks to the way we defined imitation. (Again, the relations chapter offers a much more detailed explanation of how relations may be defined.)

```
The description of a thing is usually "[comparison with
ideal][run paragraph on]".
```

```
To say comparison with ideal:
    let the sample be the ideal of the noun;
    if the sample is not a thing:
        say "Perfectly conforming to your expectations of
its type.";
        rule succeeds;
    if the material of the noun is not the material of the
sample:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun], and made of
[material of the noun].";
            otherwise say "Unusually tall at [height of the
noun], and made of [material of the noun].";
        otherwise:
            say "Distinct mostly in being made of [material of
the noun].";
    otherwise:
        if the height of the noun is not the height of the
sample:
            if the noun is shorter than the sample, say
"Unusually short at [height of the noun].";
            otherwise say "Unusually tall at [height of the
noun].";
        otherwise:
```

```
say "In every respect [a sample]."
```

```
The Pleasure Garden is a room. "At the riverbank, a
pleasing garden, having many curving paths and one
straight."
```

```
The low table is a table in the Pleasure Garden. The
height of the low table is 2 feet 3 inches. On the low
table is a metal rock called a gold nugget.
```

```
Test me with "x table / x nugget".
```

So far the effect is not very deep, but we could take the auto-description a great deal further: providing a larger and more interesting set of variations; or writing a complicated set of rules such that the player only notices height variations when carrying a ruler; or switching between several player-characters, each of whom notices a different subset of characteristics. But these refinements would require more input from later chapters.

Example **Terror of the Sierra Madre**

Multiple player characters who take turns controlling the action.

Suppose we have a game where we want the player to control two different characters, swapping bodies from one turn to the next. First, the setting, and the two people who will alternately play:

```
"Terror of the Sierra Madre"
```

```
The Hay-Strewn Corridor is a room. "[if the player is
Maleska]The horse stalls are empty: you have already
drained the animals, and carried off their corpses. The
house will not long sustain you now.
```

```
The window throws on the floor a bright square of
malevolent sunlight[otherwise]The stalls for horses run
down one side of the room, but the house has long
stood empty. A square window without shutters looks
out over the ranch, away toward the Sierras[end if]."
```

```
Teresa is a woman in the Hay-Strewn Corridor. "Teresa
stands opposite you[if Teresa carries something], her
fingers wrapped tightly around [a list of things carried by
Teresa][end if]." Teresa carries a bulb of garlic and a
cross.
```

```
Maleska is a man in the Hay-Strewn Corridor. "Maleska
watches you from eyes entirely black." Maleska carries
a skull.
```

If we tried the text above in Inform, we would find ourselves in the Hay-Strewn Corridor and confronted by both Teresa and Maleska. If "player" is not set to any named person, Inform creates a bland person called "yourself" to represent the player. To avoid this, we set "player" to the person we want to begin as. The player character is normally privately-named, so we'll need to make sure "Maleska" still means what it should.

```
The player is Maleska. Understand "Maleska" as
Maleska.
```

Now the Corridor contains just two people, and we arrive on the scene as Maleska, with only Teresa facing us.

At the end of every turn we will use the 'now the player is...' phrase. (This looks as if it simply changes the value of "player": which it does, but it also carries out a complicated operation behind the scenes to effect the switch.)

```
Every turn:
    if the player is Maleska, now the player is Teresa;
    otherwise now the player is Maleska.
```

Our two characters already see the Corridor differently, but let's

differentiate them further:

```
Every person has a number called strength. The
strength of Teresa is 3. The strength of Maleska is 5.
```

In this small example, strength is not used for anything, except that we will display it on the status line:

```
When play begins:
    now the command prompt is "[bold type][player]
[roman type] > ";
    now the left hand status line is "[player]";
    now the right hand status line is "STR: [strength of
the player]".
```

That last rule doesn't quite do what we might have expected. When we print "[player]", we find that Inform usually prints "yourself". This is because Inform says "you" to mean Teresa when talking to the player-being-Teresa, and likewise for Maleska. We want to override that in this particular story, because the rapid switches of personality are otherwise hard to follow. So:

```
Rule for printing the name of Teresa: say "Teresa".


Rule for printing the name of Maleska: say "Maleska".


Test me with "look / look".
```

Chapter 6: Commands

g Things in Directions; §6.10. Entering and Exiting, Sitting and Standing; §6.11. Waiting, Sleeping; §6.12

Contents of *The Inform Recipe Book*
Chapter 5: The Viewpoint Character
Chapter 7: Other Characters
Indexes of the examples

§6.1. Designing New Commands

Quite a bit of interactive fiction design involves the creation of custom commands to expand on the library's existing set. There is more to know than we can review in this section; instead, this is to serve as an overview of the process, with hints about where in *Writing with Inform* we might find more technical details.

Before we even start to write our source text, we should think about the following things:

> (1) What words will the player use to make this new action happen?
> (2) What will the action change about the world model?
> (3) What circumstances might make the new action go wrong or produce silly outcomes?

To take these one one by one:

(1) We may have a general idea of the phrasing we want the player to use -- say we want to add an SHOOT command which allows the player to fire a gun at something. (This is an intentionally tricky choice of verb, because it shows off so many possibilities.) So we might decide the base form of the action will be

> SHOOT THE PISTOL AT HENRY

So now we're going to need an action that applies to two objects -- the pistol as the noun, and Henry as the second noun. The problem is, though, that there are lots of other ways that the player could reasonably formulate the command, some of which

leave out information:

```
SHOOT HENRY
SHOOT PISTOL
FIRE PISTOL
SHOOT AT HENRY
SHOOT AT HENRY WITH GUN
```

To avoid frustrating the player, we should make a guess about what the player means whenever we're sure that guess will be reliable (we might, for instance, have only one gun in the story, so we know that SHOOT HENRY will always mean SHOOT HENRY WITH PISTOL), but ask the player for clarification whenever there might be ambiguity (SHOOT PISTOL gives no clue about the target, nor can we safely guess, so we want Inform to ask "What do you want to shoot the pistol at?"). The next section goes into more detail about how to handle these variations.

Conversely, there are cases where the player is offering too *much* information for the command we've defined - say we have a BURN command which doesn't look for a specified fire source, but the player is trying to BURN BOX WITH MATCH. We probably don't want to throw away the extraneous information as though it had never been typed, because the player might have typed something quite specific. BURN BOX WITH ACID, say, should not be cavalierly reinterpreted as BURN BOX (with a fire source). Instead, we want to give the player a bit of gentle guidance, perhaps using "Understand as a mistake", as in

```
Understand "burn [something] with [text]" as a mistake ("Your choice
of lighter isn't important in this story: BURN SOMETHING will
suffice.")
```

Finally, there are some cases where we want to understand a phrase to mean a specific form of a more general action. For instance, we might want TURN DOWN THE MUSIC to mean the same thing as SET VOLUME KNOB TO 1. In this case, we may want to make a sort of dummy action which converts into the main action, as in

```
Understand "turn down volume" or "turn down music" or "turn down
the volume" or "turn down the music" as lowering the volume.
Lowering the volume is an action applying to nothing.

Instead of lowering the volume, try setting the volume knob to 1.
```

More about this can be found later in this chapter, under Remembering, Converting

and Combining Actions.

Sometimes these kinds of details can be caught in play-testing, but it's a good idea to think about them specifically and in advance rather than leaving them to our beta-testers to sort out.

(2) To generalize very broadly, there are two possible kinds of command in IF: those that only exist to give the player new information (like EXAMINE, INVENTORY, LOOK, TASTE), and those that change the world model (like TAKE FISH, OPEN DOOR, UNLOCK GATE WITH BLUE KEY). The Inform library has some commands that really do none of these things by default - commands like JUMP that do nothing interesting at all most of the time - but those exist as hooks, in case there is ever something important for them to do.

Commands that ask for information are usually easier to implement. Very often we're looking to offer the player a new kind of information about specific objects, and these can be handled by adding new text properties, as in

```
A thing has some text called the sacred emanation.

Carry out perceiving something:
    say "[sacred emanation of the noun][paragraph break]".
```

Commands that affect the world model, on the other hand, can range from simple to very complex indeed. Sometimes we need to do nothing more than add an attribute to an object, like

```
A thing can be folded or flat. A thing is usually flat.
```

so that our FOLD command can change the object into its folded form. At other times, we need quite intricate rules to account for a subtle multi-stage process - how fire is burning and spreading to objects, say, or how a conversation is progressing. Other parts of the Recipe Book offer solutions to some of these challenges.

(Strictly, we might count a third kind of command: the kind that controls the story itself. The Advanced Actions chapter discusses how to add actions out of world, as these are called, but the difficult ones are already built into Inform - saving, restoring, restarting, undoing a turn, and so on. Mostly when we need to add new actions out of world, they will be help or hint systems of some kind. More about these can be found in the Helping and Hinting section of the Recipe Book, under Out of World Actions and Effects.)

(3) Most commands that change the world require certain preconditions: the player

needs to be holding the gun before he can fire it; the gun must be loaded with ammunition; if we're being especially detailed in our simulation, the safety must be off.

Often, there are also subtler details about how the command should interact with special items. For any new command we create, it's worth asking: should anything special happen if the player performs this action...

> On himself?
> On another living character?
> On an object he (or another character) is carrying or wearing?
> On an object he (or another character) is inside or on?
> On a door?
> On an object that is impossible to move (defined as "scenery" or "fixed in place")?
> On an intangible object (such as a beam of sunlight)?
> On an object far away (such as the sun)?
> On an object that is part of something else (such as a doorknob)?
> On an object that itself has parts (such as a desk with drawers)?
> If there are two objects required by the action, can both the noun and the second noun be the same thing?

For instance, we might have written code so that if the gun is fired at anything but a person or a fragile object, the default response is "The bullet bounces harmlessly off [the second noun]." Our checklist would remind us to write special cases to prevent

> SHOOT GUN AT MY SHOE (while he's wearing it)
> SHOOT GUN AT ME
> SHOOT GUN AT GUN

and so on. Actions that destroy objects are especially tricky, because there are many things that aren't safe to destroy without carefully adjusting the world model. (What happens if we burn a door connecting two rooms? a wooden desk with a drawer containing an asbestos vest? the armchair Cousin Fred is sitting on?)

Start of Chapter 6: Commands

Back to Chapter 5: The Viewpoint Character: §5.6. Viewpoint

Onward to §6.2. Writing New Commands

§6.2. Writing New Commands

Once we've considered all the design issues pertaining to a new action, we're ready to start writing the source text. First we need to give the player a way to issue the command:

```
Understand "smile" as smiling.
Understand "fold [something]" as folding.
Understand "shoot [something preferably held] at [something]" as
shooting it with.
Understand "wrap [something preferably held] in [something
preferably held]" as wrapping it in.
```

(Note how "it" stands in for the first item when we have an action requiring two objects.) The things that go in square brackets are called "tokens": they are blank spaces for the player to fill in with story objects. The different kinds of tokens are explained in the chapter on Understanding.

We can add synonyms with

```
Understand the command "grin" as "smile".
```

and we can create reversed versions of commands with

```
Understand "shoot [something] with [something preferably held]" as
shooting it with (with nouns reversed).
```

These variations are also covered in the Understanding chapter. If the action needs to work on things that aren't within the player's sight or reach in the normal way, we may need to use an [any thing] token (see the Understanding chapter), as in

```
Understand "contemplate [any thing]" as considering.
```

We may also need to modify reach or light levels (see Changing reachability and Changing visibility in the Advanced Actions chapter), or rely on the Deciding the scope of... activity.

As for guessing the player's intention when he isn't clear, we may want to consult the Does the player mean rules (to help Inform make guesses between multiple possible targets) and the activities Supplying a missing noun and Supplying a missing second

noun (to help Inform guess an appropriate item when the player leaves something entirely out of his command). For instance, if the player typed SHOOT HENRY, it is the supplying a missing noun/second noun activity that would allow us to make Inform draw the obvious conclusion that he shoots Henry with the pistol in his hand. The Does the player mean rules are discussed in the chapter on Advanced Actions; the activities in the Activities chapter.

Next we need to define our new action, as in

```
Smiling is an action applying to nothing.
Folding is an action applying to one thing.
Wrapping it in is an action applying to two carried things.
```

In cases where we're using an "[any thing]" token to let the player affect objects that aren't normally visible or reachable, we'll need to define the action to apply to *visible objects*. This tells Inform that the player doesn't have to be able to touch the object for it to work. So for instance

```
Considering is an action applying to one visible thing.
```

For more on this topic, see Visible vs. touchable vs. carried in the Advanced Actions chapter.

The next step is to create rules for Inform to follow when the action happens. These can be check rules (which make sure that the conditions for the action to occur are fulfilled); carry out rules (which perform the action); and report rules (which describe the results of the action to the player). Any new action should have at least a report rule to let the player know what has happened (if anything), and a carry out rule if there are any ramifications for the world model. For instance:

```
Carry out folding:
    now the noun is swan-like.

Report folding:
    say "You deftly fold [the noun] into the shape of a swan."
```

It's important to remember that report rules may be describing something whose name is plural, such as papers or shoes, and write our text so that it sounds right either way; see the chapter on Adaptive Text and Responses.

More about defining actions and creating carry out and report rules may be found in

the chapter on Advanced Actions.

Meanwhile, the check rules give us a chance to provide sensible restrictions on how the command works, as in

```
Check folding:
    if the noun is not a napkin:
        say "[The noun] won't bend." instead.


Check shooting something with the noun:
    say "[The noun] is incapable of aiming at itself." instead.


Check burning something which contains the player:
    say "You're not quite desperate enough to make a funeral pyre for
yourself just yet." instead.
```

The chapter on Advanced Actions explains how check rules work. In the special case where we want the player to take things automatically before using them, we may want to define the action to work only on carried objects, as in

```
Wrapping it in is an action applying to two carried things.
```

The activity Implicitly taking something (documented in the Activities chapter) allows us to modify what should happen during this process.

Lastly, a word or two about trouble-shooting. If a newly created command seems not to be working, we can discover what action Inform is really generating with the ACTIONS testing command, as in

```
>actions
Actions listing on.


>i
[taking inventory]
You are carrying nothing.
```

```
[taking inventory - succeeded]
```

If the desired command is not happening, we may need to review our understand lines. A common problem is that our new action conflicts with one already defined by default. In that case, we may want to check the Actions index and see whether there are already-defined actions which might conflict with it. If so, we may need to redefine a command with a line like

```
Understand the command "stand" as something new.
```

If that's not enough, we can get a comprehensive view of everything that happens during an action with RULES: this will list all the check, carry out, and report rules that Inform is using to perform the command.

See **Memory and Knowledge** *for more about the any token and the concept of scope to control what the player may refer to in a command*

Start of Chapter 6: Commands
Back to §6.1. Designing New Commands
Onward to §6.3. Modifying Existing Commands

§6.3. Modifying Existing Commands

Much of the rest of this chapter discusses the behavior of specific commands in Inform's command library, and how we might change and build on these. This section is instead an overview of the general principles: where and how can one intervene?

Whenever we are dealing with actions, the Actions Index is likely to be useful: it lists all the actions currently implemented, whether in our own source or in extensions or the Standard Rules, and lists the rules pertaining to each.

The lightest and easiest way to change behavior is with an Instead rule:

```
Instead of eating the apple:
    say "It turns out to be made of beeswax, so that's a non-starter."
```

```
Instead of tasting an edible thing:
    say "It's delicious!"
    rule succeeds.
```

The addition of "rule succeeds" tells Inform that the instead action was a success rather than a failure; this is not usually very important with the player's own actions, but can be useful for actions performed by other characters, so that a successfully replaced action is not followed by the disconcerting line

```
Clark is unable to do that.
```

Before and After offer alternative easy forms of modification; the Basic Actions chapter explains all three.

Changing the way an action works in all cases is usually better addressed by changing the main rulebook, rather than with one (or many) instead rules. We may add new check, carry out, and report rules to existing action rulebooks. The Advanced Actions chapter describes these, and ends with some guidelines on when to use before, instead, and after, and when to use check, carry out, and report.

Similarly, we may delete, move, or replace rules that are already present (see the chapter on Rulebooks). This is handy if we decide that an action has restrictions that we dislike and want to abolish. If the restriction we need to change is part of the accessibility rules - those which check whether the player can take, see, and touch items - we may need to look at Changing reachability or Changing visibility in the Advanced Actions chapter (to revise what is allowed), at Deciding the scope of something in the Activities chapter (to influence what can be seen when).

If, for instance, the player character is a burly fellow who can lift any other character he likes:

```
The can't take other people rule is not listed in any rulebook.
```

...and rip knobs off doors:

```
The can't take component parts rule is not listed in the check taking
rulebook.
```

...and commit petty theft:

```
The new can't take people's possessions rule is listed instead of the
can't take people's possessions rule in the check taking rulebook.


This is the new can't take people's possessions rule:
    if someone (called the owner) carries the noun:
        say "(first waiting until [the owner] is distracted)";
```

The right approach to use also depends a bit on how systematic a change we anticipate. We may find that instead rules become cumbersome when we want to specify behavior for a very large number of objects. It's fine to have

```
Instead of tasting the arsenic:
    say "You'll live to regret this very very shortly.";
    end the story.
```

but a bit more tedious to have to write

```
Instead of tasting the peppermint: ...
Instead of tasting the plate: ...
Instead of tasting the banister: ...
Instead of tasting the donkey: ...
(etc.)
```

in a story in which most items have unique flavor descriptions. In that situation, it may be more sensible to overhaul the design of the action: create a new text property for things, and revise "tasting" so that it now consults this property:

```
The block tasting rule is not listed in any rulebook.


A thing has some text called the flavor. The flavor of a thing is usually
"Nothing special."


Report tasting something:
    if the flavor of the noun is "Nothing special.":
        say "You taste nothing unexpected." instead;
    otherwise:
```

```
say "[the flavor of the noun][paragraph break]" instead.


Report someone tasting something:
    say "[The actor] licks [the noun]."
```

Finally and most sweepingly, we can rip out whole passages of the Standard Rules and replace them - or not. This is a drastic measure and rarely necessary (or so we hope); but see the Extensions chapter for ways to replace sections of existing source, or even revise the Inform 6 template files on which Inform depends. By these means almost anything can be changed. We can throw out a whole range of existing commands and start from scratch, for instance, if we want Inform to know about a completely new and different command set.

See **Magic (Breaking the Laws of Physics)** *for a hat that lets the player walk through closed doors, and an NPC able to reach through solid containers*

Start of Chapter 6: Commands

Back to §6.2. Writing New Commands

Onward to §6.4. Looking

Example 214: **Slogar's Revenge** Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

§6.4. Looking

Looking is quite a complicated command, since the production of a room description takes many steps. A detailed description of this process may be found in the Room Descriptions section.

By convention, a player sees full descriptions of rooms he enters more than once, but may type BRIEF in order to see shorter descriptions, and SUPERBRIEF tells the story never to print room descriptions at all. VERBOSE restores the default behavior.

These conventions are not always appropriate, however, especially in works where experiencing a changing environment is essential. The use option

Use brief room descriptions.

changes the default behavior so that rooms are not always described fully to the player. **Verbosity** demonstrates how this works.

The player always has the option of turning room descriptions to BRIEF or SUPERBRIEF mode. **Verbosity 2** demonstrates how we might remove the player's ability to change the default behavior.

See **Room Descriptions** *for a detailed description of how Inform creates room descriptions and how to change the results*

See **Going, Pushing Things in Directions** *for ways to change just those room descriptions that are shown as the result of the player's movement*

See **Memory and Knowledge** *for ways to change the room description in response to the player character's knowledge at any given stage of play*

Start of Chapter 6: Commands

Back to §6.3. Modifying Existing Commands

Onward to §6.5. Examining

Example 3: **Verbosity 1**
Making rooms give brief room descriptions when revisited.

Example 395: **Verbosity 2**
Making rooms give full descriptions each time we enter, even if we have visited before, and disallowing player use of BRIEF and SUPERBRIEF.

---

§6.5. Examining

By default, examining an object shows its description, and - for devices - tells us whether the object is switched on or switched off.

This kind of additional information is not always what we want, so if we have a device whose on/off status we want to conceal, we may write

The examine described devices rule is not listed in any rulebook.

On the other hand, there are times when we may want to add a similar line or two to the descriptions of other kinds of objects. **Crusoe** allows us to append an "It is charred." sentence to the end of descriptions of things we have burned in the fire. Since it works by introducing a "printing the description" activity, Crusoe is also a good example to start from if we want to introduce more complex, flexible descriptions of items throughout our story.

Odin rewrites the "You see nothing special..." line with other text of our own, for items that otherwise do not have a description.

Finally, we may want to look at multiple things at once. **The Left Hand of Autumn** demonstrates how we might provide a different response for EXAMINE PAINTINGS than for examining each individually; **Beekeeper's Apprentice** provides a SEARCH command that will show the descriptions of all the scenery in the current location.

See **Actions on Multiple Objects** *for an alternative EXAMINE ALL command*

Start of Chapter 6: Commands

Back to §6.4. Looking

Onward to §6.6. Looking Under and Hiding

Example 44: ★ **Odin** Replacing "You see nothing special..." with a different default message for looking at something nondescript.

Example 68: ★★ **Beekeeper's Apprentice** Making the SEARCH command examine all the scenery in the current location.

Example 295: ★★★ **The Left Hand of Autumn** The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple

examination command that describes groups in special ways.

Example 337: ★★★ **Crusoe** Adding a "printing the description of something" activity.

---

§6.6. Looking Under and Hiding

Finding hidden objects is a classic puzzle in IF. **Beachfront** provides the most basic example, an object that becomes visible only when we have searched the papers on a cluttered desk. **Beneath the Surface** takes this further, giving all large furnishings the ability to conceal items, and allowing the player to put things underneath other things, as well as find them. **Flashlight** adds an extra twist to the puzzle by requiring that the player have a flashlight to shine under a bulky object in order to find what lies underneath.

Looking inside an object is generally handled by the searching action, and we could extend that to allow the player to search multiple or complex objects. **Matreshka** turns the puzzle on its head by allowing the player to search a whole room systematically with only a single command.

★ *See* **Kitchen and Bathroom** *for the related case of needing to look in a mirror*

Start of Chapter 6: Commands

Back to §6.5. Examining

Onward to §6.7. Inventory

Example 98: ★ **Beachfront** An item that the player can't interact with until he has found it by searching the scenery.

Example 172: ★ **Matreshka** A SEARCH [room] action that will open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

Example 218: ★ **Flashlight** Visibility set so that looking under objects produces no result unless

the player has a light source to shine there (regardless of the light level of the room).

Example 233: **Beneath the Surface** An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

---

§6.7. Inventory

Occasionally we would like to change the way the name of something is printed as part of our inventory, and we can do this with a printing the name rule such as

```
Rule for printing the name of the dead rat while taking inventory:
    say "dead rat (at arm's length)"
```

There are also several possibilities for redesigning the inventory list as a whole. **Persephone** shows how to divide an inventory list into two parts, a "You are carrying: " section and a "You are wearing: " section. **Equipment List** goes further, and shows how we might use Inform's specialized listing functions to create a variety of differently formatted inventories.

Sometimes the way Inform by default lists properties such as "(closed)" or "(open but empty)" isn't quite what we want. **Oyster Wide Shut** offers a flexible alternative to the standard behavior, allowing finer control over which properties are listed and how they are described.

Trying Taking Manhattan replaces the inventory behavior for other characters: instead of silently looking through their possessions (but not describing them to the player), they now describe to the player what they're carrying and wearing.

Start of Chapter 6: Commands

Back to §6.6. Looking Under and Hiding

Onward to §6.8. Taking, Dropping, Inserting and Putting

Example 425: **Oyster Wide Shut** Replacing Inform's default printing of properties such as

"(closed)", "(open and providing light)", etc., with our own, more flexible variation.

Example 177: ★★ **Equipment List** Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

Example 65: ★★★ **Persephone** Separate the player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

Example 204: ★★★ **Trying Taking Manhattan** Replacing the inventory reporting rule with another which does something slightly different.

---

§6.8. Taking, Dropping, Inserting and Putting

We may want to change the default refusal message when the player tries to pick up scenery: **Replanting** demonstrates this case simply.

Removal modifies responses to successful TAKE commands, with the effect that when the player picks up an item, he gets a response such as "You take the book from the shelf."

Croft modifies the DROP command, so that objects dropped on specific surfaces get reported in a special way. **Celadon** allows the player to drop even objects he is carrying indirectly, for instance on a tray or in a sack.

Morning After introduces a simple rule that changes the behavior of the whole story: whenever the player takes an item he hasn't already looked at, he automatically examines it. This picks up the pace of exploration passages where the player is likely to be collecting a large number of objects.

By default, when the player tries to put or insert an item that he isn't holding, Inform

prints a refusal message; **Democratic Process** and **Sand** offer ways instead to have the player first pick up the relevant items. (The former applies to single items the player is trying to place; the latter expands coverage to work even if the player uses a command affecting multiple objects.)

Taking also happens as a result of other commands. Such takes can be made unnecessary by turning off the "carrying requirements rule" under particular circumstances, or presented differently using the implicitly taking activity.

Start of Chapter 6: Commands

Back to §6.7. Inventory

Onward to §6.9. Going, Pushing Things in Directions

Example 16: ★ **Replanting** Changing the response when the player tries to take something that is scenery.

Example 90: ★ **Morning After** When the player picks something up which he hasn't already examined, the object is described.

Example 198: ★ **Removal** TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

Example 224: ★ **Celadon** Using the enclosure relation to let the player drop things which he only indirectly carries.

Example 86: ★★★ **Democratic Process** Make PUT and INSERT commands automatically take objects if the player is not holding them.

Example 202: ★★★ **Croft** Adding special reporting and handling for objects dropped when the player is on a supporter,

and special entering rules for moving from one supporter to another.

Example 379: ★★★ **Lollipop Guild** Overriding the rules to allow the player to show something to another character without first taking it.

Example 87: ★★★★ **Sand** Extend PUT and INSERT handling to cases where multiple objects are intended at once.

---

§6.9. Going, Pushing Things in Directions

Going is the most complex of actions after looking (or perhaps including looking): the success of every movement depends on the direction the player goes; the room he starts from; the room he intends to reach; whether there are any doors intervening (and, if so, whether these are closed or locked); whether he is traveling by vehicle; and whether he is pushing anything in front of him. When he gets there, the description he sees is itself generated by a looking command.

Pushing something in a direction is really a sort of going. The command >PUSH WHEELBARROW WEST first checks certain qualifying rules: by default, only things defined as pushable between rooms may be pushed, and they may be pushed only in horizontal directions (not UP or DOWN) -- though these rules can be overridden, as we see in **Zorb**. If the player's pushing attempt passes these criteria, the action is translated automatically into a going action, with all the usual checks about whether that direction leads anywhere, whether a door is in the way, and so on. The converted action afterward can be caught with such rules as

```
Instead of going to the Alpine Meadow with the wheelbarrow:
    say "You don't want to crush the delicate blooms."


Instead of going north with the handcart:
    say "The headwind is so stiff that you are unable to make much
northerly progress at all while encumbered by the handcart."
```

Since the two actions are internally being handled as one, both are discussed here.

It is very common for players to make a mistake and type the wrong direction command, or even to misunderstand the room description and not recognize all the possible exits. **Bumping into Walls** helpfully adds a facility so that when the player tries to go in the wrong direction, the story lists the correct possibilities, as in

```
From here, the viable exits are to the south, the east and the west.
```

Assuming that travel succeeds, another useful technique is to provide some sense of the journey between locations, especially if they are remote from one another or the player has to do something unusual to get from one to the other. **Up and Up** adds a short description of travel when we approach a new room, before the room description is printed; **Veronica**, conversely, adds a comment when the player leaves a region of the map. **The Second Oldest Problem** intervenes and kills a player who tries to travel from one dark room to another. **Mattress King** embellishes the description that automatically results from PUSH MATTRESS WEST, adding a line that describes the player pushing the object before describing the new room approached.

We may also want to add a brief comment when we arrive in a new room, after the room description is printed. One trivial way to do this is to append the line to the room's main description, conditionally, like this:

```
The Hammock Emporium is a room. "This is Cousin Ed's shop, the
big dream he left accounting to pursue. You can't help gawking at the
Luxury Leather Space Hammock, made of genuine red buffalo
skins[if unvisited]. [paragraph break]So this is why Grampa makes all
those 'lying down on the job' jokes every Thanksgiving[end if].".
```

But often we want our first-glance comment to come after some items in the room are described; and for this effect, we would use the "first look rule" defined in **Saint Eligius**.

If these methods are not enough, the looking action has an action-specific variable called "the room-describing action", which records whether this particular instance of looking comes about because the player typed LOOK or because the player traveled to a new location. We can consult this variable if we want to make looking work differently after going, as for instance here:

```
Check looking when the room-describing action is the going action:
    say "You are temporarily too blinded to see." instead.
```

Another category of examples treat how we handle the movement commands themselves. The eight compass directions, with UP and DOWN, IN and OUT, are

used as standard in most interactive fiction, but they are not the only possible way of navigating, and strike many newcomers to the genre as counter-intuitive, since when strolling around in real life most of us rarely think about our travel in terms of compass orientation. **Misadventure** allows the player to GO TO a named room, instead, and calculates the best route to reach the destination; **Safari Guide** builds on this by letting the player make the whole trip in a single move, automatically opening any doors that stand in his way en route.

In the same spirit of interpreting the player's intentions sensibly, **Provenance Unknown** modifies the pushing command so that if the player pushes the top object in a stack of objects towards a direction, Inform attempts to move the bottom item instead. This is convenient if, for instance, we have a heavy television on a movable cart and want PUSH TELEVISION WEST to work just as well as PUSH CART WEST.

We also sometimes want to respond sensibly to terse movement commands or ones that rely on some knowledge of where the player has already been. **Polarity** provides a GO BACK command, allowing the player to retreat in the direction from which he came, while **Minimal Movement** understands LEAVE, GO, and so on as OUT, in the absence of other information. **Owen's Law** takes this further, calculating from the best routes on a map how to make OUT mean "move towards the exit of this indoor room", and IN mean "proceed further into the interior". **Wonderland** assigns altitudes to all rooms and works out the local best meaning of UP and DOWN accordingly.

See **Map** *for how to create other kinds of new direction*

See **Varying What Is Read** *for further divisions of the standard compass, such as north-northwest*

See **Ships, Trains and Elevators** *for ship-board directions*

See **Bicycles, Cars and Boats** *for common vehicles in which to travel the map*

Start of Chapter 6: Commands

Back to §6.8. Taking, Dropping, Inserting and Putting

Onward to §6.10. Entering and Exiting, Sitting and Standing

Example 100: **Veronica** An effect that occurs only when the player leaves a region entirely.

Example 105: **Mattress King** Adding extra phrasing to the action to PUSH something in a direction.

Example 200: ★ **The Second Oldest Problem** Adapting the going action so that something special can happen when going from a dark room to another dark room.

Example 306: ★ **Misadventure** A going by name command which does respect movement rules, and accepts names of rooms as commands.

Example 372: ★ **Minimal Movement** Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

Example 393: ★ **Saint Eligius** Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

Example 6: ★★ **Up and Up** Adding a short message as the player approaches a room, before the room description itself appears.

Example 256: ★★ **Wonderland** Hiking Mount Rainier, with attention to which locations are higher and which lower than the present location.

Example 307: ★★ **Safari Guide** The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling

all openable doors on the way.

Example 102: ★★★ **Bumping into Walls** Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

Example 103: ★★★ **Polarity** A "go back" command that keeps track of the direction from which the player came, and sends him back.

Example 107: ★★★ **Provenance Unknown** Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

Example 108: ★★★ **Zorb** Replacing the message the player receives when attempting to push something that isn't pushable, and also to remove the restriction that objects cannot be pushed up or down.

Example 179: ★★★ **Owen's Law** OUT always means "move to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

---

§6.10. Entering and Exiting, Sitting and Standing

Under ordinary circumstances, Inform does not keep track of the player's posture, nor of his exact location in a room. **Lies** implements a room in which the player can lie in different positions on the floor, getting different views as a result.

Our other examples are all modifications of the way Inform handles player movement to make better default guesses at what he wants to do: **Anchorite** adds a GET DOWN and DOWN command that work when the player is on a supporter, to

accompany GET UP, GET OFF, and GET OUT (already understood). **Get Axe** makes the player get out of a portable container before attempting to lift it - a consideration that comes up relatively rarely, but that might pertain to inflatable rafts, beanbag chairs, and other lightweight but capacious pieces of furniture.

See **Position Within Rooms** *for a box the player can push around the room and stand on in different locations*

See **The Human Body** *for letting the player sit, stand, or lie down systematically on furniture or on the floor*

See **Furniture** *for various objects on which the player can sit or stand*

Start of Chapter 6: Commands

Back to §6.9. Going, Pushing Things in Directions

Onward to §6.11. Waiting, Sleeping

Example 206: **Get Axe** Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

Example 290: **Anchorite** By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.

Example 310: **Lies** Commands to allow the player to lie down in three different ways.

---

§6.11. Waiting, Sleeping

The standard WAIT command makes time pass at the same rate that it would anyway

- one minute per turn. In a story where events happen at specific times of day, though, we might want to give the player more control. **Nine AM Appointment** shows how to give the player a WAIT 10 MINUTES command, while **Delayed Gratification** lets him WAIT UNTIL a specific time of day.

Ordinarily, Inform also refuses to allow the player to SLEEP and WAKE UP: the commands exist, but have no effect. **Change of Basis** lets the player put himself into a sleep state in which he cannot do anything. A somewhat more interesting expansion on this idea would be to let the player sleep and have dreams; there are no examples specifically of dream states, but we might consult the examples on scenes about how to disrupt one environment and move the player to another, entirely new one.

See **Scene Changes** *for ways to move the player to a new environment such as a dream state*

Start of Chapter 6: Commands

Back to §6.10. Entering and Exiting, Sitting and Standing

Onward to §6.12. Other Built-In Actions

Example 388: **Nine AM Appointment** A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

Example 47: **Change of Basis** Implementing sleeping and wakeful states.

Example 389: **Delayed Gratification** A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

---

§6.12. Other Built-In Actions

Many other actions are themselves very simply implemented and provide only a shell for us to expand on according to the needs of a particular story. Many of these are discussed at more length in sections on various kinds of props and objects; in particular:

See **Modifying Existing Commands** *for ways to override automatic takes or restrictions on what the player must be able to hold or touch*

See **Sounds** *for LISTEN*

See **Barter and Exchange** *for GIVE and SHOW*

See **Combat and Death** *for ATTACK*

See **Saying Simple Things** *for ASK, TELL, and ANSWER*

See **Food** *for TASTE and EAT*

See **Liquids** *for DRINK*

See **Clothing** *for WEAR and TAKE OFF*

See **Bags, Bottles, Boxes and Safes** *for OPEN, CLOSE, LOCK, and UNLOCK as applied to containers*

See **Doors, Staircases, and Bridges** *for OPEN, CLOSE, LOCK, and UNLOCK as applied to doors*

See **Furniture** *for things the player can ENTER and GET OUT of*

See **Money** *for BUY*

See **Fire** *for BURN*

See **Glass and Other Damage-Prone Substances** *for CUT*

Start of Chapter 6: Commands
Back to §6.11. Waiting, Sleeping
Onward to §6.13. Magic Words

---

§6.13. Magic Words

Many fantasy games incorporate the idea of magic words that can be spoken. In implementing these, we want to be a bit flexible and accept a range of input regardless of whether the player explicitly speaks the command aloud: XYZZY, SAY XYZZY, or perhaps even CAST XYZZY. The inventively named **Xyzzy** demonstrates

how we might define such a command.

If we want to go even further and to allow the player also to use quotation marks, as in SAY "XYZZY", we may want to include Punctuation Removal by Emily Short, which allows for quotation marks to be stripped out of the player's input before it is understood.

Start of Chapter 6: Commands

Back to §6.12. Other Built-In Actions

Onward to §6.14. Remembering, Converting and Combining Actions

Example 287: ★ **XYZZY** Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

---

§6.14. Remembering, Converting and Combining Actions

Sometimes we want Inform to apply a player's action to a different target than the one specified: for instance, directing all (or almost all) commands from the doorknob to the door of which it is a part. **Fine Laid** demonstrates how to do this. Along the same lines, **Lucy** shows how to direct a player's conversation action to apply to a new conversation topic.

We can also record a series of actions performed by the player or by another character.

Cactus Will Outlive Us All demonstrates characters each of whom reacts to a very specific provocation; **I Didn't Come All The Way From Great Portland Street** implements a game show in which the player is not allowed ever to repeat an action he has already performed; and **Leopard-skin** implements a maze which the player can escape only by performing a specific sequence of actions.

Anteaters provides a peculiar gizmo that can remember actions performed in its presence and force the player to reiterate them.

Start of Chapter 6: Commands

Back to §6.13. Magic Words

Onward to §6.15. Actions on

Multiple Objects

Example 88: ★ **Fine Laid** Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

Example 92: ★ **Lucy** Redirecting a question about one topic to ask about another.

Example 220: ★ **Cactus Will Outlive Us All** For every character besides the player, there is an action that will cause that character to wither right up and die.

Example 430: ★ **Leopard-skin** A maze that the player can escape if he performs an exact sequence of actions.

Example 435: ★ **I Didn't Come All The Way From Great Portland Street** In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

Example 222: ★★ **Anteaters** The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing

actions that apply to topics, as in "look up anteater colonies in the guide".

§6.15. Actions on Multiple Objects

Inform allows a handful of actions - TAKE, DROP, PUT, INSERT - to apply to more than one item at a time, so that the player can move things around easily.

The general principle is that multiple objects are allowed if the actions are likely to be successful but not interesting most of the time, and if they're things that the player could plausibly do all at once. For most actions, the use of ALL would seem weirdly indiscriminate: EAT ALL, say, describes very implausible behavior, and EXAMINE ALL would likely generate a screenful of text at once.

But this is all under our control. To create an action that uses multiples, or to allow the use of multiple objects with an already-existing action, we need to create an understand statement that uses the "[things]" token (note the plural). For instance:

```
Understand "give [things] to [someone]" as giving it to.
```

This would let the existing give action apply to multiple objects, in just the same way that "take" does. **Shawn's Bad Day** demonstrates how we might allow EXAMINE ALL to print descriptions of every visible item.

Alternatively, we could generate a new action:

```
Understand "give [things] to [someone]" as multiply-giving it to.
Multiply-giving it to is an action applying to one carried thing and one
thing.
```

(In theory the language here should perhaps be "several carried things" -- but Inform is still going to process multiply-giving item by item, unless we redirect it. More about this in a moment.)

When handling an action that uses the "[things]" token, the parser makes a list of every item to which it is going to apply the action: this is called the multiple objects list. The multiple objects list can be the result of a vague request (GET ALL) or a specific one involving identical multiples (GET PENNIES, GET THREE APPLES) or a very specific one involving unique, named nouns (GET GERBIL, APPLE, AND POMEGRANATE).

We can manipulate what Inform includes in "ALL" in sentences like TAKE ALL with

the "deciding whether all includes..." activity; for instance

```
Rule for deciding whether all includes scenery: it does not.
```

prevents TAKE ALL from applying to things that can't be moved anyway, avoiding lots of lines like

```
tree: That's hardly portable.
swing set: That's hardly portable.
```

A slightly tedious technical note: the multiple objects list is not strictly a list in the standard Inform sense, because it is used so frequently in parsing that it would be cumbersome to handle it with the more flexible but less efficient structure used for lists. However, if we want to manipulate the multiple objects list as though it were an ordinary list -- that is, sort it, rotate it, truncate it, remove entries from it, etc -- we may do so by creating a list like this:

```
let L be the multiple object list.
```

and later after making L conform to our desires:

```
alter the multiple object list to L.
```

Inform next repeatedly runs the action rulebook for the action generated, using each item from the multiple object list as "noun" in turn (or as "second noun", if that's where the [things] token appeared in the understand line). Since it is possible to alter the multiple object list before the "generate action rule" portion of the turn sequence consults the rulebooks, we can also affect the order in which the player's matched objects are handled; see **Formicidae**. We should not attempt to change the multiple object list after this point, because this is likely to introduce bugs.

Each time Inform tries the action on a new noun, it prefixes the action-attempt with the name of the item it's currently working on. This is where we get such output as "frog eyeballs:" and "newt toes:" in long lists like

```
frog eyeballs: Taken.
newt toes: Taken.
```

These names are generated by the "announce items from multiple object lists rule" in the action-handling rules; **Escape from the Seraglio** shows how to alter them. In the context of this rule, the thing we are currently printing the name of can be called "the current item from the multiple object list".

Suppressing names of objects entirely, while occasionally tempting, may have unintended consequences, especially if some of the attempted actions are prevented by check rules that themselves print things. It is safest to suppress the multiple object names in the case where we already know that the action will succeed wherever it is attempted (more often for observational actions like examining than for manipulative actions like taking, or where we mean to completely override default handling).

Given that our hypothetical "multiply-giving" applies to each given object in turn, it might seem to be useless to create "multiply-giving" as an action different from "giving" -- but the convenience is that manipulating the multiple object list makes it possible to group behavior artificially. The trick here is that, on the first pass of the multiply-giving rulebook, we look at the entire multiple object list, perform actions, print output, and set a flag saying that the action has been handled. The flag tells Inform not to do or print anything for any of the subsequent passes through that action rulebook; thus we artificially create a situation where, instead of performing an action on each object in turn, Inform acts once on the entire group. That allows us to assess the cumulative qualities of the group and have the action respond differently than it might when assessing each item individually.

The Facts Were These demonstrates how we might write an action for GIVE THREE DOLLARS TO MAN or GIVE PIE AND HAT TO MAN where the man would only accept the collective gift when its total proved satisfactory.

Western Art History 305 demonstrates how we might allow EXAMINE, which doesn't normally permit multiple objects, to take them, but to give vaguer responses to a mass examination than an individual one.

See **Examining** *for groups of objects that have a collective description different from their individual descriptions, and for commands that search multiple things at once*

See **Dispensers and Supplies of Small Objects** *for ways to let the player pick up a number of identical items from a dispenser or supply*

Start of Chapter 6: Commands

Back to §6.14. Remembering, Converting and Combining Actions

Onward to §6.16. Alternate Default Messages

Example 294: **Shawn's Bad**

Day Allowing the player to EXAMINE ALL.

Example 327: ★★ **The Best Till Last** Reordering multiple objects for dramatic effect.

Example 328: ★★ **Western Art History 305** Allowing EXAMINE to see multiple objects with a single command.

Example 411: ★★ **Escape from the Seraglio** Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

Example 428: ★★ **Formicidae** Manipulating the order in which items are handled after TAKE ALL.

Example 431: ★★ **The Facts Were These** Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

---

§6.16. Alternate Default Messages

Often we will want to replace the text produced by Inform by default: this includes quite a wide range of text, much of which either describes the success of a command or explains why the action failed.

Inform provides the Responses system to enable default messages like "You can't go that way" to be changed, and this is capable of making large-scale changes. This is especially useful if we want to give the viewpoint character a distinctive voice and set of mannerisms.

Start of Chapter 6: Commands

Back to §6.15. Actions on Multiple Objects

Onward to §6.17. Clarification and Correction

§6.17. Clarification and Correction

Some commands and some objects raise special challenges when it comes to working out the player's intention.

Sometimes this can be done with good rules about the assumptions Inform should make. **Alpaca Farm** demonstrates a USE command, always a challenge because USE can mean very different actions with different items.

There are also times when we need to ask the player for more information. **Apples** demonstrates how sensibly to use properties to disambiguate between similar objects, while **Walls and Noses** rephrases the disambiguation question when special objects are involved: examining one of the walls of the room will make the story ask "In which direction?" and EXAMINE NOSE will lead to "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

At other times, the player types something that is wrong in a predictable way: for instance, we might want to remove all the "with..." phrases from commands like

```
HIT DOOR WITH FIST
KICK DRAGON WITH FOOT
LOOK WEST WITH EYES
```

and merely parse the remainder of the command. (That last command may be unlikely, but novice players do quite often type commands that refer unnecessarily to body parts.) **Cave-troll** demonstrates how.

WXPQ demonstrates how to modify the error message the parser gives in response to a command it doesn't understand; this particular example focuses on the "That noun doesn't make sense in this context" message that arises from using the "[any thing]" or "[any room]" tokens, but the techniques could be adapted to handling other parser errors as well.

For catching typing errors, Cedric Knight's extension Mistype may also be of use: it provides an automatic typo-correction function that the player can turn on or off.

Start of Chapter 6: Commands

Back to §6.16. Alternate Default Messages

Onward to §6.18. Alternatives To Standard Parsing

Example 289: ★ **Alpaca Farm** A generic USE action which behaves sensibly with a range of different objects.

Example 368: ★ **Apples** Prompting the player on how to disambiguate otherwise similar objects.

Example 380: ★ **WXPQ** Creating a more sensible parser error than "that noun did not make sense in this context".

Example 370: ★★★ **Walls and Noses** Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

Example 424: ★★★ **Cave-troll** Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

---

§6.18. Alternatives To Standard Parsing

Very occasionally, for out-of-the-ordinary games, we want to make major changes to

the way that Inform ordinarily understands commands.

Cloves shows how we might read adverbs in the player's command: adverbs are challenging because they can legitimately appear anywhere in a command structure, so must be found and accounted for before the rest of the command is understood.

Fragment of a Greek Tragedy goes further, substituting a keyword-recognition parser for the usual structure of commands and objects.

Less drastically, menus of numbered options can temporarily replace or augment standard commands. **Down in Oodville** demonstrates how to add a list of transporter destinations from which the player may choose by numeral.

See **Traits Determined By the Player** *for ways to ask the player a question at the beginning of play*

See **Saying Simple Things** *for a way to ask the player a yes-no question any time during play*

Start of Chapter 6: Commands

Back to §6.17. Clarification and Correction

Onward to Chapter 7: Other Characters: §7.1. Getting Acquainted

Example 304: **Down in Oodville** Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

Example 373: **Cloves** Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

Example 374: **Fragment of a Greek Tragedy** Responding to the player's input based on keywords only, and overriding the

Examples from Chapter 6: Commands

Start of this chapter
Chapter 7: Other Characters
Indexes of the examples

214 **Example** **Slogar's Revenge** *WI*

Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

Under most circumstances, locking and unlocking require the player to be carrying the key he uses to unlock something. This makes sense -- unless the key is on a keychain, or on a chain around his neck, for instance. So here we explore one way to circumstantially override the carrying requirements, while still making sure that the player cannot unlock the door if the unlocking tool is nowhere in sight.

In essence, we are rewriting the carrying requirements rule with a different one of our own devising, and swapping it in only at those moments when it is correct to do so.

```
"Slogar's Revenge"


Section 1 - Procedure


The amulet carrying rule substitutes for the carrying
requirements rule when locking something with the
Amulet of Tumblers.


The amulet carrying rule substitutes for the carrying
requirements rule when unlocking something with the
Amulet of Tumblers.
```

We can now replace the usual behavior of the carrying requirements rule (to check whether the player is carrying something and, if not, to generate an implicit take) with a similar rule of our own; note that "if the player has the second noun" is a more compact way to write "if the player carries the second noun or the player wears the second noun":

```
This is the amulet carrying rule:
    if the player has the second noun:
        continue the action;
    say "(first picking up the amulet)[command
clarification break]";
    try silently taking the second noun;
    if the player is not carrying the second noun:
        stop the action;



Section 2 - Scenario



The Daunting Dungeon is a room.



West of the Daunting Dungeon is the Disturbing Door.
The Disturbing Door is a door. West of the Disturbing
Door is the Fallow Field.



The Disturbing Door is closed and locked.



The player wears the Amulet of Tumblers. The Amulet
of Tumblers unlocks the Disturbing Door.



Test me with "unlock disturbing door with amulet / open
door / west / remove amulet / close door / lock
disturbing door with amulet / drop amulet / unlock
disturbing door with amulet".
```

For a more systematic handling of the keychain problem (and a number

of other refinements to the behavior of doors), see the Locksmith extension included with Inform.

Making rooms give brief room descriptions when revisited.

By default, the description of a room is printed every time the player enters a room.

On a device with very limited screen space, however, we might wish to supplant that behavior with "brief" descriptions. In Brief mode, Inform prints room descriptions only when the player enters that room for the first time. Afterwards, the text is skipped, for brevity, though the player can see it again at any time by typing LOOK.

As we saw in the previous chapter, we can set "use options" to control certain aspects of the player's experience. One of the use options is the option to

```
Use brief room descriptions.
```

which changes the defaults so that the description of a room is printed only the first time the player enters.

```
"Verbosity"

Use brief room descriptions.

The Wilkie Memorial Research Wing is a room. "The
research wing was built onto the science building in
1967, when the college's finances were good but its
aesthetic standards at a local minimum. A dull brown
corridor recedes both north and south; drab olive doors
open onto the laboratories of individual faculty
members. The twitchy fluorescent lighting makes the
whole thing flicker, as though it might wink out of
```

existence at any moment.

The Men's Restroom is immediately west of this point."

The Men's Restroom is west of the Research Wing.
"Well, yes, you really shouldn't be in here. But the nearest women's room is on the other side of the building, and at this hour you have the labs mostly to yourself. All the same, you try not to read any of the things scrawled over the urinals which might have been intended in confidence."

Test me with "west / east".

If we type "test me" during play, these commands will be carried out automatically, and we can see that when we return to the Research Wing, the description is not given a second time.

Some notes: the player can also turn full-length descriptions on or off with the commands "verbose" and "brief", or set a minimal-description setting with the command "superbrief". This power still belongs to the player even if we have set the use option to show brief room descriptions by default.

Moreover, we can ourselves check what the state of the descriptions is, with

```
if set to sometimes abbreviated room descriptions: ...
if set to unabbreviated room descriptions: ...
if set to abbreviated room descriptions: ...
```

Finally, it is possible to exercise more precise control over what the player sees on his first and subsequent visits to a room; see the next example for details.

ExampleVerbosity 2
Making rooms give full descriptions each time we enter, even if we have visited before, and

Suppose that we want the player always to see full room descriptions, even if he tries to reset the defaults -- perhaps because there is vital information there which he will miss if he turns off full-length room descriptions.

To do this, we might want to remove the standard behavior of the three actions associated with BRIEF, SUPERBRIEF, and VERBOSE, replacing them with explanatory messages about how the game behaves. We cannot use Instead to override these actions, because Instead rules do not apply to actions out of world. Instead, we will want to remove and replace the carry out rules.

We can do this easily by going to the Actions Index, looking up the detail panel for, say, "preferring abbreviated room descriptions", and click the "unlist" button to paste in the sentence that will remove this rule from the rulebook.

Let's remove all three of the carry out rules and substitute our own:

```
"Verbosity 2"

Section 1 - Procedure

The prefer unabbreviated room descriptions rule is not
listed in the carry out preferring unabbreviated room
descriptions rulebook.

The prefer sometimes abbreviated room descriptions
rule is not listed in the carry out preferring sometimes
abbreviated room descriptions rulebook.

The prefer abbreviated room descriptions rule is not
listed in the carry out preferring abbreviated room
descriptions rulebook.
```

Carry out preferring unabbreviated room descriptions:
    say "[story title] always provides full-length descriptions for your reading pleasure."

Carry out preferring sometimes abbreviated room descriptions:
    say "For your playing protection, [story title] provides only full-length room descriptions."

Carry out preferring abbreviated room descriptions:
    try preferring sometimes abbreviated room descriptions instead.

The standard report preferring abbreviated room descriptions rule is not listed in the report preferring abbreviated room descriptions rulebook.

The standard report preferring unabbreviated room descriptions rule is not listed in the report preferring unabbreviated room descriptions rulebook.

The standard report preferring sometimes abbreviated room descriptions rule is not listed in the report preferring sometimes abbreviated room descriptions rulebook.

Use full-length room descriptions.

Section 2 - Scenario

The Wilkie Memorial Research Wing is a room. "The research wing was built onto the science building in 1967, when the college's finances were good but its aesthetic standards at a local minimum. A dull brown corridor recedes both north and south; drab olive doors open onto the laboratories of individual faculty members. The twitchy fluorescent lighting makes the whole thing flicker, as though it might wink out of existence at any moment.

The Men's Restroom is immediately west of this point."

The Men's Restroom is west of the Research Wing. "Well, yes, you really shouldn't be in here. But the nearest women's room is on the other side of the building, and at this hour you have the labs mostly to yourself. All the same, you try not to read any of the things scrawled over the urinals which might have been intended in confidence."

Test me with "west / east / brief / w / e / superbrief / w / e / verbose".

ExampleOdin

Replacing "You see nothing special..." with a different default message for looking at something nondescript.

In recent years there has been a strong trend towards providing unique descriptions for all implemented objects. Often this is a good idea, but there are also contexts in which we may want to discourage the player from looking too closely at some things and concentrate his attention on just a few interesting ones.

The trick here is that leaving items completely undescribed leads to rather dull exchanges like this:

```
>x table
You see nothing special about the table.
```

...which can leave the player with the impression that the author was simply too lazy to describe everything. So it can be a good idea to replace that default message with a different one more appropriate to the game. For instance:

```
"Odin"

The House of a Mortal Farmer is a room. "Having two
separate rooms, this house testifies to considerable
wealth and success at agriculture."

The Bedroom is inside from the House.

A chair is a kind of supporter. A chair is always
enterable.

In the House are a table, two chairs, a basket, and a
hearth. On the table is a loaf of bread.

The description of a thing is usually "You give [the noun]
a glance, but it is plainly beneath your attention."
```

Because the description is attached to a whole kind ("thing"), it is really a blanket instruction about many objects at once. More specific instructions always override less specific ones, so we can easily make exceptions. For instance, the following will work correctly:

```
The infant is a man in the basket. The description of the
infant is "So strong and fat that you wonder whether one
of your fellow gods is acquainted with the mistress of
the house-- but it's no concern of yours, of course."
```

Test me with "x table / x chair / x infant".

Example **Beekeeper's Apprentice**

Making the SEARCH command examine all the scenery in the current location.

We have to create a suitable action and say what it does, and to repeat what we do through all the scenery items. That needs material from subsequent chapters, but is quite ordinary Inform all the same:

```
"Beekeeper's Apprentice"

Studying the vicinity is an action applying to nothing.

Report studying the vicinity:
    if the location does not contain something which is
scenery:
        say "There's little of interest in the [location]."
instead;
    repeat with point of interest running through scenery
in the location:
        say "[point of interest]: [run paragraph on]";
        try examining the point of interest.

Understand "search" as studying the vicinity.

The Yard is a room.

The hive and the honey are scenery things in the Yard.
```

```
The description of the hive is "The honeycombed hive is
all around you, thrumming with life." The description of
the honey is "Wax-sealed honey has been cached in
many of the hexagonal nurseries."



Test me with "search".
```

The reason for this example is to show the use of saying "[run paragraph on]". It means we have output such as:

```
>search
hive: The honeycombed hive is all around you,
thrumming with life.



honey: Wax-sealed honey has been cached in many of
the hexagonal nurseries.
```

Without the running on, the prompts "hive:" and "honey:" would be separated from the descriptions following them, which would look a little odd.

---

295 **Example** **The Left Hand of Autumn** *WI*

The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple examination command that describes groups in special ways.

Suppose that we have a game in which groups of objects can have meaning apart from their individual significance -- perhaps there are spells that can only be cast by collecting just the right items in the same place.

In this case, one of the things the player might like to be able to do is look at several items together and get a special response, different from looking at the items individually.

To make this happen, we need to do several things:

(1) we need to create a version of the EXAMINE command that can apply to multiple objects at once.
(2) we need to correct the way Inform normally deals with multiple-object commands, because we want our group description to print only one time, and we want to avoid stubs such as "pear: ... apple: ..." before or after the group description.
(3) we need to define a way for Inform to identify interesting groups and describe them.

"The Left Hand of Autumn"

Section 1 - Procedure

Understand "examine [things]" or "look at [things]" as multiply-examining. Multiply-examining is an action applying to one thing.

Understand "examine [things inside] in/on [something]" or "look at [things inside] in/on [something]" as multiply-examining it from. Multiply-examining it from is an action applying to two things.

Group-description-complete is a truth state that varies.

Carry out multiply-examining it from:
    try multiply-examining the noun instead.

Check multiply-examining when group-description-complete is true:
    stop the action.

```
Carry out multiply-examining:
    let L be the list of matched things;
    if the number of entries in L is 0, try examining the
noun instead;
    if the number of entries in L is 1, try examining entry 1
of L instead;
    describe L;
    say line break;
    now group-description-complete is true.


Before reading a command:
    now group-description-complete is false.
```

Now for step 2, overriding Inform's usual output of names of objects:

```
The silently announce items from multiple object lists
rule is listed instead of the announce items from multiple
object lists rule in the action-processing rules.


This is the silently announce items from multiple object
lists rule:
    unless multiply-examining or multiply-examining
something from something:
        if the current item from the multiple object list is not
nothing, say "[current item from the multiple object list]:
[run paragraph on]".


Definition: a thing is matched if it is listed in the multiple
object list.
```

We'll save our "to describe" phrase until Section 2, when we can give the game specific instructions about how to report different lists of objects.

Now, the player might also want to be able to refer to a group of item by some kind of group name, so let's add the option of creating a Table of Collective Names which will interpret these:

```
After reading a command:
    repeat through the Table of Collective Names:
        let N be "[the player's command]";
        let Y be relevant list entry;
        while N matches the regular expression "[name-
text entry]":
            replace the regular expression "(.*)[name-text
entry](.*)" in N with "\1[Y]\2";
        change the text of the player's command to N.



Report taking something:
    say "You pick up [the noun]." instead.
```

And as a bit of polish, because we'd like SEARCH TABLE to have the same effect as EXAMINE ALL ON TABLE:

```
Understand "look on [something]" as searching.



Instead of searching something which supports at least
two things:
    let L be the list of things supported by the noun;
    describe L.
Instead of searching something which contains at least
two things:
    let L be the list of things contained by the noun;
    describe L.



Section 2 - Scenario



Eight-Walled Chamber is a room. "A perfectly octagonal
room whose walls are tinted in various hues."



The display table is a supporter in the Chamber. A twig
of rowan wood is on the table.
```

```
The player carries an apple and a pear.
```

```
A glove is a kind of thing. A glove is always wearable.
Understand "glove" as a glove. The player carries a left
glove and a right glove. The left glove and the right
glove are gloves.
```

Now we define a few actual lists of items:

```
Fruit list is a list of objects which varies. Fruit list is {
apple, pear }.
Glove list is a list of objects which varies. Glove list is {
right glove, left glove }.
Arcane list is a list of objects which varies. Arcane list is
{ left glove, twig, pear }.
```

```
To describe (L - a list of objects):
    sort L;
    if L is fruit list:
        say "Just a couple of fruits.";
    otherwise if L is glove list:
        say "It's a matched pair of fuzzy blue gloves.";
    otherwise if L is arcane list:
        say "To anyone else it might look like a random
collection of objects, but these three things -- [L with
definite articles] -- constitute a mystic key known as the
Left Hand of Autumn. They practically hum with power.";
    otherwise:
        say "You see [L with indefinite articles]."
```

```
When play begins:
    sort fruit list;
    sort glove list;
    sort the arcane list.
```

We sort the lists so that regardless of how we change the rest of the code (and the order in which objects are coded), the resulting list will always be in sorted order and ready to compare with the list of items the player wants to look at. And thanks to the "Reading a command" code we wrote earlier, we can also teach the game to understand the player's references to "the left hand of autumn" as a specific collection of items.

Table of Collective Names

relevant list	name-text
"[orchard list]"	"fruits of autumn"
"[glove list]"	"gloves"
"[glove list]"	"pair of gloves"

Test me with "x apple and pear / x left and right / put pear on table / put left glove on table / x all on table / put all on table / examine all on table / get apple, twig, pear / x all on table / search table".

---

337 ExampleCrusoe

Wi

Adding a "printing the description of something" activity.

Suppose we want to add rules so that any time we examine a charred object (and most of our objects can be charred), a line about the charring is appended to the end of the object description. We could use "after examining...", but perhaps we would prefer for the sentence about the charring not to appear in its own paragraph.

This is an ideal occasion for a new activity. We look at the action index for "examining" to identify the rule that causes the old behavior (in this case, the "standard examining rule"); replace this with a new rule that calls our activity; and write our "printing the description" activity in such a way that it uses an object's description without forcing a paragraph return afterward.

Then we will use "after printing the description" to add our line about charring, and make sure that the paragraph return does occur before the prompt.

So:

"Crusoe"

Section 1 - Creating our New Activity

```
The fancy examining rule is listed instead of the
standard examining rule in the carry out examining
rules.
```

This instruction replaces a normal piece of the examine action, the standard examining rule, with another one of our own devising. (The replacement of the standard examining rule will be explained in more detail in the chapter on rulebooks.)

```
Printing the description of something is an activity.
```

```
This is the fancy examining rule:
    carry out the printing the description activity with the
noun;
    rule succeeds.
```

All we have done here is enclose what is usually just a rule inside an activity. This means that we can now write before and after rules for the activity, and also add special instructions like "Rule for printing the name of something while printing the description of something" -- this may not be likely to arise often, but Inform now has the concept of "printing the description of something" as a separate context of action. Next we add the modification that lets us append to the description without a new line:

```
Rule for printing the description of something (called
item):
    if the description of the item is not "":
        say "[description of item] [run paragraph on]";
    otherwise:
        say "You see nothing special about [the item]. [run
paragraph on]".
```

"run paragraph on" here will mean that we do not get a paragraph break following the description, even if it ends with a period. We also insert a space, so that our follow-on comments will be properly punctuated.

```
After printing the description of something charred:
    say "It is charred." instead.
```

The instead at the end of this line stops Inform for going on with any other "after printing the description of..." rules.

The standard library also has rules for printing additional text about containers and supporters with visible contents, and devices that are switched on; with this current system, we could add those as "after printing the description" rules as well, building up a complete paragraph if we wanted. But for simplicity we won't exemplify all of that here. The effects would be much the same as with the "charred" line.

Now, because we want to make sure that we always do get a paragraph break after our description, we add this rule last after all the other rules. "Last" and "first" rules are covered in more detail in the chapter on rulebooks.

```
Last after printing the description of something:
    say paragraph break.
```

Section 2 - The Scenario

```
The Desert Isle is a room. "A pale expanse of sand,
here and there developing into hillocks of grass, and a
small clump of palms. The water is shallow here, and
there are other islands within swimming distance -- or
even wading distance, perhaps -- but none of them is
any larger than your island, so it doesn't seem worth the
trouble of visiting.

A few hundred feet out, the water turns darker blue, the
sea floor drops away, and there is nothing to be seen all
the way down to the horizon, except a couple of fluffy
clouds, and an occasional bird.
```

```
The remains of your fire smolder in the stone-lined pit."
```

```
A thing can be charred or whole. A thing is usually
whole. Instead of burning something: say "You hold [the
noun] to the fire until it flares and chars."; now the noun
is charred.
```

```
The player carries a stick. The description of the stick is
"A strip of palm from the woodiest part of the leaf, about
a foot and a half long."
```

```
The player carries a glass bottle and a piece of paper.
The description of the paper is "A single blank sheet." In
the glass bottle is a grain of sand. The glass bottle is
openable and open. Instead of burning the glass bottle:
say "You hold the bottle to the flame, but it grows
uncomfortably warm."
```

```
Instead of burning the grain of sand: say "You drop the
grain into the fire pit, where it becomes indistinguishable
from all the others."; now the grain of sand is nowhere.
Instead of dropping the grain of sand: now the grain of
sand is nowhere; say "You return the grain of sand to its
brethren."
```

The player's description is handled in an unusual way, and this will produce a space paragraph break there where it should not. Instead, therefore, we will add an instead for examining the player (probably a good idea anyway):

```
Instead of examining the player:
    say "You are sunburned and there is sand in cracks
you didn't know existed."
```

```
Test me with "i / x stick / x bottle / x sand / x paper / x
me / burn stick / x stick / burn paper / x paper".
```

The "printing a description" activity may be useful for other games, and can be imported just by lifting section 1.

ExampleBeachfront

An item that the player can't interact with until he has found it by searching the scenery.

Suppose we have our player, a detective, searching for evidence; we don't want him to be able to use this evidence until he has performed the action that reveals it, but after that it should be visible in the room when he looks.

A simple way to do this is to start the object -- an envelope, in this scenario -- out of play, and only move it into the location when the player looks for it:

"Beachfront"

The Stuffy Office is a room. "The windows are closed, making the sultry air even more unbearable. A narrow slice of Caribbean blue is visible between the scuba gear rental shop and the recreated 17th century pirate tavern.

The office is cheerfully furnished with wicker chairs and white curtains, but the tropical decorating scheme stopped at the desk, which is heavy oak and absolutely covered with papers."

The heavy oak desk is a supporter in the stuffy office. It is scenery. Understand "paperwork" as the desk.

The creamy envelope is an openable container. The description is "There is no return address on the outside

```
of the envelope, just the address of the Doctor's office --
but the legs of the capital A are rubbed down in a
characteristic way, and the top of every R is open.
There's no question that it comes from the same
typewriter as the blackmail note." In the envelope is a
letter. The envelope can be found or lost. The envelope
is lost.


Instead of searching the desk when the envelope is lost:
    now the envelope is found;
    say "You rifle through the piles of bills and notices;
invitations to conventions; advertisements for high-end
prescription drugs; pink carbon sheets bearing
patients[']  names and medical identification numbers in
spidery, elderly handwriting. Almost at the bottom of the
heap, you find what you were looking for: a creamy
envelope with the address typed.";
    move the envelope to the desk.
```

Here we've changed the property of the envelope to keep track of the fact that it has been found, so that if the player tries again, he won't find anything more.

```
Instead of searching the desk:
    say "Further investigation of the desk reveals nothing
else suspicious."
```

Notice that we have two rules that apply to "searching the desk", but one of them has a more specific set of parameters ("when the envelope is lost"). This means that Inform will consult that rule first and use it if it applies; it will only carry out our plain vanilla "instead of searching the desk" rule when the more restricted rule is not relevant.

```
Test me with "x envelope / x desk / search desk / look /
get envelope / x envelope".
```

ExampleMatreshka

A SEARCH [room] action that will open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

"Matreshka"

Ransacking is an action applying to one thing.

Check ransacking:
    if the noun is not the location, say "You can hardly search [the noun] from here." instead.

Carry out ransacking:
    while the player can see a closed openable unlocked container (called target):
        say "[target]: [run paragraph on]";
        try opening the target.

Report ransacking:
    say "You can see nothing further worth searching."

The Russian Gift Shop is a room. In the Russian Gift Shop is a large wooden doll. It is closed and openable. In the large wooden doll is a medium wooden doll. It is closed and openable. In the medium wooden doll is a small wooden doll. It is closed and openable. In the small wooden doll is a tiny solid wooden doll.

And now we need to borrow from a later chapter for the command that will make this work:

Understand "search [any visited room]" as ransacking.

Test me with "search gift shop".

Visibility set so that looking under objects produces no result unless the player has a light source to shine there (regardless of the light level of the room).

```
"Flashlight"


The Schoolhouse is a room. "Though normally
comfortable, the room is dark and menacing during the
storm; rain sheets on the windows, and you can barely
see the flash of the lighthouse only a few miles away."


The cabinet is a fixed in place openable container in the
Schoolhouse. The hurricane lantern is a thing in the
Schoolhouse. "A hurricane lantern hangs from a peg."
The lantern is lit.


Visibility rule when looking under something:
    if the player is carrying a lit thing (called lamp):
        say "You shine [the lamp] under [the noun]...";
        there is sufficient light;
    there is insufficient light.


There is a marble. The marble can be found or lost. The
marble is lost.


Instead of looking under the cabinet when the marble is
lost:
    move the marble to the player;
    now the marble is found;
    say "Billy's lost marble! So that's where it got to!"
```

Test me with "look under cabinet / get lantern / look under cabinet".

Because visibility is checked before instead rules, this discovery will (correctly) occur only when the player does have enough light.

Example **Beneath the Surface** *WI*

An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

The standard world model provides for the idea of containers and supporters, but this is not the only way that objects can relate to one another in the real world. Here we try adding the idea of concealment beneath another object:

```
"Beneath the Surface"

Section 1 - In Which our Terms are Defined

Underlying relates various things to one thing. The verb
to underlie means the underlying relation. The verb to
be under means the underlying relation. The verb to be
beneath means the underlying relation.

Instead of looking under a thing which is underlaid by
something (called the lost object):
    say "You find [the list of things which underlie the
noun]!";
    now every thing which underlies the noun is carried
by the player;
    now every thing which underlies the noun does not
underlie the noun.

Hiding it under is an action applying to one carried thing
```

and one thing. Understand "put [something preferably held] under [something]" as hiding it under. Understand "hide [something preferably held] under [something]" as hiding it under. Understand the commands "shove" and "conceal" and "stick" as "hide".

Check hiding it under:
    if the second noun is not fixed in place, say "[The second noun] wouldn't be a very effective place of concealment." instead.

Carry out hiding it under:
    now the noun is nowhere;
    now the noun underlies the second noun.

Report hiding it under:
    say "You shove [the noun] out of sight beneath [the second noun]."

Section 2 - In Which They are Put To Use

The Room of Hidden Objects is a room. It contains a sofa, an easy chair, and a rug. The sofa supports a lime-green pillow and an innocent-looking Chinese finger toy. The rug is fixed in place. The chair is a supporter.

A treasure map underlies the easy chair. A skeleton is beneath the sofa. A blueprint of Atlantis, a lexicon of Linear A, and the key to Jimmy Hoffa's Mausoleum are under the rug.

Test me with "look under the sofa / look under the rug / look under the easy chair / hide lexicon under rug".

Example Oyster Wide Shut

Replacing Inform's default printing of properties such as "(closed)", "(open and providing light)", etc., with our own, more flexible variation.

As we've seen in earlier examples such as "Equipment List", it is possible to vary the way Inform creates inventory listings in general (to create lists that look more like paragraphs of prose, lists divided between what the player is wearing and what he isn't, and so on). We can also use activities to alter the printing of specific objects' names and contents, as with the "omit contents in listing" feature after printing the name of something.

We may find, however, that we would like a great deal more control over Inform's printing of inventory details, not just as a special effect for a few items, but throughout the game.

We start by turning off Inform's native property writer:

```
"Oyster Wide Shut"

Section 1 - Procedure

The print standard inventory rule is not listed in any
rulebook.

Carry out taking inventory (this is the new print inventory
rule):
    say "You are carrying: [line break]";
    list the contents of the player, with newlines,
indented, including contents, with extra indentation.
```

This is very much like the library's standard behavior, but with the exception that "giving inventory information" or even "giving brief inventory information" are omitted. Here's how we supplant it:

```
After printing the name of something (called target)
while taking inventory:
    follow the property-aggregation rules for the target.
```

Now, our property-aggregation rulebook is going to look at a given object and decide on a list of features that should be mentioned in inventory. We'll start by producing something quite similar to Inform's default behavior:

```
The property-aggregation rules are an object-based
rulebook.
The property-aggregation rulebook has a list of text
called the tagline.


A first property-aggregation rule for an openable open
thing (this is the mention open openables rule):
    add "open" to the tagline.


A first property-aggregation rule for an openable closed
thing (this is the mention closed openables rule):
    add "closed" to the tagline.


A property-aggregation rule for a closed transparent
container which contains nothing (this is the mention
empty transparent containers rule):
    add "empty" to the tagline.


A property-aggregation rule for an open container which
contains nothing (this is the mention empty open
containers rule):
    add "empty" to the tagline.


A property-aggregation rule for a lit thing (this is the
mention lit objects rule):
    add "providing light" to the tagline.
```

```
A property-aggregation rule for a thing worn by the
player (this is the mention worn objects rule):
    add "being worn" to the tagline.


The last property-aggregation rule (this is the print
aggregated properties rule):
    if the number of entries in the tagline is greater than
0:
        say " ([tagline])";
        rule succeeds;
    rule fails.
```

Notice that we don't need to write any rules about how to print that list of text: because Inform is printing out a list, it will automatically insert commas, spaces, and the word "and" where appropriate; and it will automatically follow the "use serial comma" option, if we have it set.

Now we're free to meddle. Let's give the player a bunch of possessions that will be listed in interesting ways in inventory:

```
Section 2 - Scenario


The Curved Beach is a room. "White sand stretches
away both northeast and northwest, enclosing this
attractive little bay. Gentle waves lap at the beach."


The player carries a glowing plastic sack. The glowing
plastic sack is lit and transparent and openable and
open. It contains a rock. It is wearable.


The player wears a flashlight lanyard. The flashlight
lanyard is a device.


Carry out switching on the lanyard: now the noun is lit.
```

```
Carry out switching off the lanyard: now the noun is
unlit.
```

```
The player carries an oyster. The oyster contains a
pearl. The oyster is openable.
```

Now suppose that we don't want the oyster to say "closed" when it's closed. Instead, we'd like it to say "clamped shut". As this is the only property the oyster will ever have, we can simply override his whole property-aggregation rulebook:

```
A property-aggregation rule for the oyster:
    if the oyster is closed:
        say " (clamped shut)";
    rule succeeds.
```

That's fine for the oyster because "clamped shut" is the only property he'll ever have. What if we'd like instead just to revise the way the sack (and only the sack) gets described as providing light?

```
The sacklight rule is listed after the mention lit objects
rule in the property-aggregation rules.
```

```
A property-aggregation rule for the plastic sack (this is
the sacklight rule):
    if "providing light" is listed in the tagline:
        remove "providing light" from the tagline;
        add "gently glowing" to the tagline.
```

Now the flashlight (and any other regular light sources we might add to the game) will be described as "providing light", but the sack will only be said to be gently glowing -- a bit more appropriate for its rather fainter gleam.

We might also wish to add a systematic feature across the board to include a new property in the inventory list? Let's say the player can enchant his possessions, and enchanted possessions should thereafter be listed accordingly:

```
A thing can be magical or non-magical. A thing is
usually non-magical.

Understand "enchant [something]" as enchanting.
Enchanting is an action applying to one thing.

Carry out enchanting something:
    now the noun is magical.

Report enchanting something:
    say "Ding! You turn [the noun] magical."

A property-aggregation rule for a magical thing:
    add "enchanted" to the tagline.

Test me with "i / close oyster / i / turn on flashlight / i /
take off flashlight / i / turn off flashlight / i / close sack / i /
open sack / i / take all from sack / i / close sack / i / wear
sack / i / enchant sack / i / open sack / put all in sack / i /
close sack / i".
```

Further variations are possible as well: if we used a "before printing the name..." rather than an "after printing the name..." rule, we could automatically generate lines like "an open and empty phosphorescent plastic sack", removing some of the artificiality of the parentheses.

Or we could add more logic to the rules about which properties are mentioned, so that some features of objects were mentioned in inventory only if the player was wearing the correct detection device, like so:

```
The player wears enchantment-detecting goggles.

A property-aggregation rule for a magical thing:
    if the player is wearing the goggles:
```

add "enchanted" to the tagline.

Example **Equipment List**

Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

Most of the phrase options above are relatively self-explanatory; a few are less so. Here is an overview:

"With newlines" tells Inform to put a new line before each listed object. Indented tells it to indent contents of objects, when listing these.

"Giving inventory information" means to append information such as (closed) or (being worn) to objects.

"As a sentence" means to put "and" before the last object and commas between them; this is usually not used in conjunction with newline listing. "As a sentence" obeys whatever conventions about the use of the serial comma we may have established with the "Use serial comma" option.

"Including contents" means to list the contents of open or transparent containers and all supporters, whereas including all contents means to list the contents of all containers, even opaque closed ones.

"Tersely", perhaps unexpectedly, puts parentheses around objects listed as the contents of other objects.

"Giving brief inventory information" omits most of the inventory tags, such as "(open)" and "(worn)", but does list "(closed)" for closed containers which might not otherwise be obviously openable.

"Using the definite article" means prefixing objects with "the", if applicable, rather than "a".

"Listing marked items only" means including only objects that have already been declared "marked for listing".

"Prefacing with is-are" means that Inform will write "is" before the list if it contains only one item, and "are" if the list contains more than one.

"Not listing concealed items" means to omit from the list anything which is scenery.

Finally, "with extra indentation" means that the whole list should be indented slightly, in emulation of the default inventory listing.

With this information, we can try rewriting the inventory behavior to emulate the standard or to explore alternate versions:

```
"Equipment List"

The Watery Room is a room. The player carries a
snorkel and a waterproof sack. The waterproof sack
contains an undersea map, a diving guide, a cup, and
500 Argentine pesos. The cup contains a worm. The
player wears a swimsuit and a pair of flippers. The sack
is openable and open.

Inventory listing style is a kind of value. The inventory
listing styles are tall, wide, curt, minimal, divided tall,
and divided wide. Current inventory listing style is an
inventory listing style that varies.

Understand "inventory [inventory listing style]" as
requesting styled inventory. Requesting styled inventory
is an action applying to an inventory listing style. It is an
action out of world.

Carry out requesting styled inventory:
    now current inventory listing style is the inventory
listing style understood.

Report requesting styled inventory:
    say "Inventory listing is now set to [current inventory
listing style]."
```

We begin by emulating the standard inventory listing style:

```
Instead of taking inventory when current inventory listing
style is tall:
```

```
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying: [line break]";
    list the contents of the player, with newlines,
indented, giving inventory information, including
contents, with extra indentation.
```

Here we offer the alternative of listing everything together as a paragraph:

```
Instead of taking inventory when current inventory listing
style is wide:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying ";
    list the contents of the player, giving inventory
information, as a sentence, including contents;
    say "."
```

This may be unsatisfactory, however. Items that are inside other items are not set off from those merely carried by the player. One way around this is to use terse listing, giving such descriptions as "a waterproof sack (in which are an undersea map, a diving guide, a cup (in which is a worm) and a 500 Argentine pesos)" as opposed to the more confusing " a waterproof sack (open), inside which are an undersea map, a diving guide, a cup, inside which is a worm and a 500 Argentine pesos".

```
Instead of taking inventory when current inventory listing
style is curt:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are carrying ";
    list the contents of the player, tersely, giving brief
inventory information, as a sentence, including contents;
    say "."
```

If, using the above style, we close the sack, we will still get "(closed)" after the sack's listing. The following minimalist listing style abolishes even that nicety:

```
Instead of taking inventory when current inventory listing
style is minimal:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
```

```
    say "You are carrying ";
    list the contents of the player, tersely, as a sentence,
including contents;
    say "."
```

If we want to list worn things separately from carried things, we have occasion to put "listing marked items only" to work:

```
Instead of taking inventory when the current inventory
listing style is divided wide:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    say "You are wearing ";
    now all things enclosed by the player are unmarked
for listing;
    now all things worn by the player are marked for
listing;
    if no things worn by the player are marked for listing,
say "nothing";
    otherwise list the contents of the player, as a
sentence, listing marked items only;
    say ".[paragraph break]";
    say "You are carrying ";
    now all things carried by the player are marked for
listing;
    now all things worn by the player are unmarked for
listing;
    if no things carried by the player are marked for
listing, say "nothing";
    otherwise list the contents of the player, as a
sentence, tersely, giving brief inventory information,
listing marked items only;
    say ".[paragraph break]".
```

And similarly for a tall divided inventory:

```
Instead of taking inventory when the current inventory
listing style is divided tall:
    if the number of things enclosed by the player is 0,
say "You are empty-handed." instead;
    if the player carries something:
        now all things enclosed by the player are
unmarked for listing;
        now all things carried by the player are marked for
listing;
        say "You are carrying: [line break]";
```

```
        list the contents of the player, with newlines,
indented, giving inventory information, including
contents, with extra indentation, listing marked items
only;
    if the player wears something:
        now all things enclosed by the player are
unmarked for listing;
        now all things worn by the player are marked for
listing;
        say "You are wearing: [line break]";
        list the contents of the player, with newlines,
indented, including contents, with extra indentation,
listing marked items only.
```

```
Test me with "i / inventory wide / i / inventory curt / i /
close sack / i / open sack / inventory minimal / i / close
sack / i / open sack / inventory divided wide / i /
inventory divided tall / i / drop all / i / take all / take off
swimsuit / take off flippers / i / i divided wide / i / wear
swimsuit / drop all / i".
```

ExamplePersephone

Separate the player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

If we wanted, we might replace the rule for taking inventory as follows:

```
"Persephone"
```

```
Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying
[a list of things carried by the player][otherwise][We]['re]
empty-handed";
    say "[if the player wears something]. [We]['re] wearing
[a list of things worn by the player][end if]."
```

```
The Fancy Party is a room. The player carries a sword,
a strawberry stem, and 20 credits worth of platinum. The
player wears a sash indicating lordhood.


Test me with "i / take off sash / i / drop all / i".
```

ExampleTrying Taking Manhattan

Replacing the inventory reporting rule with another which does something slightly different.

Inform has built-in commands for other people, and sometimes we may want to adjust the way these work without completely disabling and replacing the command. Suppose, for instance, that instead of

```
Kermit the Frog looks through his possessions.
```

we'd like someone taking inventory to report what he's got, thus:

```
Kermit the Frog says, "I seem to be carrying a
microphone and wearing a hat and a trenchcoat."
```

To do this, we could replace the built-in report rule with a different one.

```
"Trying Taking Manhattan"


The loud inventory rule is listed instead of the report
other people taking inventory rule in the report taking
inventory rules.


This is the loud inventory rule:
    unless the player is the person asked:
        say "[The person asked] says, 'I seem to be
```

carrying [a list of things carried by the person asked][if the person asked is wearing something] and wearing [a list of things worn by the person asked][end if].'"

Persuasion rule for asking someone to try doing something: persuasion succeeds.

Grand Central Station is a room. "Here you are in New York, New York. Any minute now someone is going to burst into song."

Kermit the Frog is a man in Grand Central Station. "Kermit the Frog stands nearby, enjoying being green." Kermit is wearing a hat and a trenchcoat. He is carrying a microphone.

Test me with "inventory / kermit, inventory".

ExampleReplanting

Changing the response when the player tries to take something that is scenery.

By default, "TAKE OAK" in the example above will produce the response "That's hardly portable." This is fine under many circumstances, but also a bit generic, so we might want to override it for a specific game.

"Replanting"

The Orchard is a room. "Within this quadrille of pear trees, a single gnarled old oak remains as a memory of centuries past." The gnarled old oak tree is scenery in the Orchard.

```
Instead of taking some scenery: say "You lack the hulk-
like strength."


Test me with "take oak".
```

Here we've used an "instead" rule; we will learn more about these in the section on actions. This allows us to define our own results for taking an object.

Note: "scenery" is a property of an object (about which we will hear more later). So when we use it in rules, we can talk about "some scenery", "something that is scenery", or even "a scenery thing" -- the last one doesn't sound much like English, but is a more plausible construction with other adjectives.

ExampleMorning After

When the player picks something up which he hasn't already examined, the object is described.

Suppose we want to make the player's life slightly easier by examining everything he picks up, if he hasn't already examined it.

```
"Morning After"


A thing can be examined or unexamined.


After taking something unexamined:
    say "Taken. [run paragraph on]";
    try examining the noun.


Carry out examining something:
```

```
now the noun is examined.
```

Carry out rules are explained in more detail in the chapter on advanced action handling. For now, it may be enough to know that what we put into this carry out rule for examining will happen any time anything is examined, but that it will not interfere with the rest of the predefined behavior of the action. The player will still see the object description and so on, as usual.

```
The Red Door Saloon is a room. "This old place is in
pretty bad shape since the mine shut down. Now there's
not much to see but the pair of deep gouges in the
floorboards where they dragged away the Sheriff's
corpse with the spurs still on."

Jed is a man in the Red Door Saloon. "At 8:30 AM the
only person around is old Jed, collecting his hangover
cure."

The pistol is a thing in the Red Door Saloon. The
description of the pistol is "It ain't too accurate, but for
two dollars you can't expect much."

The hangover cure is a thing in the Red Door Saloon.
The description of the hangover cure is "Two yellow
egg-yolks unbroken in a red-brown liquid. Yep."

Test me with "x pistol / get all".
```

TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

Suppose that we want to change the reporting of "take" so that the player is always told something like "You take the book from the shelf." or "You pick up the toy from the ground." In order to generate these reports, we will need to know where the object started, even though by the time we are printing the output, the object will have moved.

```
"Removal"

The Pharmacy is a room. A desk and a trash can are in
the Pharmacy. The pill-counter, a prescription, and a
computer are on the desk. The computer is fixed in
place. The pill-counter contains some Vicodin. The trash
can contains an empty box.

The taking action has an object called previous locale
(matched as "from").
```

The previous locale could in theory be either a thing or a room, so we make it "an object" -- that is, the most generic possible kind, to which both things and rooms belong. Now we record what the previous locale is at the beginning of each taking action:

```
Setting action variables for taking:
    now previous locale is the holder of the noun.

Report taking something from the location:
    say "You pick up [the noun] from the ground."
instead.

Report taking something:
    say "You take [the noun] from [the previous locale]."
instead.

Test me with "get all".
```

Example Celadon

Using the enclosure relation to let the player drop things which he only indirectly carries.

By default, Inform only lets the player drop those things which he is carrying -- that is, those directly in his possession. Things inside satchels or on portable trays have to be taken first.

If we want to change this behavior, we might add a dropping rule that distinguishes between carrying and mere enclosure (introduced back in "The location of something" in the chapter on Things):

```
"Celadon"

The Tea Room is a room. The player carries a black
lacquer tray. The lacquer tray is portable. On the
lacquer tray are a celadon teapot and a napkin.

Before dropping something:
    if the player does not carry the noun and the player
encloses the noun:
        say "(first taking [the noun] from [the holder of the
noun])[command clarification break]";
        silently try taking the noun;
        if the player does not carry the noun, stop the
action.

Instead of taking the napkin:
    say "It seems to be stuck to the tray, possibly by an
underlying wad of gum."

Test me with "i / drop teapot / i / look / drop teapot / drop
napkin / i / drop tray".
```

Make PUT and INSERT commands automatically take objects if the player is not holding them.

"Stop" and "Continue" are most useful when we need to write rules that will have to stop the action some of the time but at other times let it pass; so for instance:

```
"Democratic Process"

Before inserting something which is not carried by the
player into something:
    if the noun is in the second noun, say "Already done."
instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

Before putting something which is not carried by the
player on something:
    if the noun is on the second noun, say "Already
done." instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

The Assembly Room is a room. "On most days, this
room is used for elementary school assemblies; at the
moment, it serves as a voting place." The ballot is on
the desk. The desk is in the Assembly Room.

The machine is a container in the Assembly Room. "On
the ballot machine is a sign which reads 'PUT BALLOTS
IN ME :)'." Understand "ballot machine" as the machine.
```

Test me with "put ballot in machine".

 ExampleCroft *WI*

Adding special reporting and handling for objects dropped when the player is on a supporter, and special entering rules for moving from one supporter to another.

Suppose that we have a design in which the player spends lots of time on enterable supporters, and in which we want to report certain actions -- dropping things onto those supporters, or leaping from one to another -- in a new way. We might begin by adding some action variables to help us keep track of the situation:

```
"Croft"

The dropping action has an object called the container
dropped into (matched as "into").

The dropping action has an object called the supporter
dropped onto (matched as "onto").

Rule for setting action variables for dropping:
    if the actor is in a container (called C), now the
container dropped into is C;
    if the actor is on a supporter (called C), now the
supporter dropped onto is C.

Report dropping a heavy thing onto a metallic thing:
    say "You drop [the noun], and [the supporter dropped
onto] clangs protestingly." instead.
```

Report someone dropping a heavy thing onto a metallic thing:
    say "[The actor] drops [the noun] onto [the supporter dropped onto], which clangs protestingly." instead.

A thing can be heavy or light. A thing can be metallic or ordinary. A thing is usually ordinary. A thing is usually light.

The Ancient Cambodian Temple is a room. "A vast space built for ancient and forgotten rituals. The stone floor crawls with vermin. Well above the floor, and separated by some feet, are twin platforms built into the wall: the one carved of jointed wood, the other of sheets of graven bronze."

A platform is a kind of supporter. A platform is always enterable. A platform is usually scenery.

The bronze platform is a metallic platform in the Temple. Lara is a woman. She is on the bronze platform. She wears safari pants and a tank top. She carries a gun and a map. The gun is heavy.

The wood platform is an ordinary platform in the Temple. The player is on the wood platform. The player carries a rope, an Ancient Cambodian/English Phrasebook, a pickaxe, and a precious idol. The idol and the pickaxe are heavy.

Persuasion rule: persuasion succeeds.

The entering action has an object called the place left (matched as "from").

```
    Check entering a platform from a platform:
        if actor is the player, say "You leap into midair to
    cross the distance...";
    otherwise say "[The actor] leaps gracefully across the
    distance...";
    move the actor to the holder of the noun, without
    printing a room description.
```

Because this rule occurs before the "implicitly pass through other barriers rule", that rule will not occur when we move from platform to platform; we'll use our own custom rule instead.

```
Rule for setting action variables for entering:
    now the place left is the holder of the actor.


Report entering a platform from a platform:
    say "You land in a cat-like crouch on [the noun]."
    instead.


Report Lara entering a platform from a platform:
    say "Lara lands soundlessly on [the noun][if the noun
    supports the player] beside you[end if]." instead.


Report entering a platform from the location:
    say "You jump, catch the edge of [the noun] in your
    hands, and -- exerting considerable upper-body strength
    -- pull yourself up onto it." instead.


Report Lara entering a platform from a location:
    say "Lara jumps, catches the edge of [the noun], and
    is standing upright on it, all in less time than it takes to
    tell."


Instead of examining a person who is not the player:
    say "[The noun] carries [list of things carried by the
    noun] and wears [list of things worn by the noun]."
```

Instead of climbing a platform, try entering the noun.

Test me with "Lara, drop map / lara, drop gun / drop idol / enter bronze platform / drop pickaxe / get off / climb wood".

ExampleLollipop Guild

Overriding the rules to allow the player to show something to another character without first taking it.

As mentioned in this section, the "implicitly taking" activity does not allow us to skip an implicit take entirely. In order to do this, we need to borrow from the chapter on Rulebooks and tell Inform that one of the rules normally built in to the Standard Rules does nothing in certain circumstances:

"Lollipop Guild"

The carrying requirements rule does nothing when showing something to the guardian.
The can't show what you haven't got rule does nothing when showing something to the guardian.
The block showing rule does nothing.

Candyland is a room. "A fizzing, popping wonderland of sugary delights. A path tiled with butterscotch sweets leads to the horizon."

The butterscotch path is scenery in Candyland.

The player carries a basket. In the basket are a licorice gumdrop and a can of tuna. The gumdrop is edible. The description of the gumdrop is "Covered all over with grains of sugar." The can of tuna is edible. The description of the can of tuna is "A rare import in this place."

The giant lollipop is a fixed in place edible thing in Candyland. "Growing right next to the path, on a trunk of white paper, is a giant lollipop colored green and red and white." The description of the lollipop is "If you were very blind, like Aunt Myrtle, you might mistake it for a young sapling just planted: the lollipop is just that leafy shade of green, with swirls of white and red that might be branches or flowers."

The guardian is a man in Candyland. "Right beside you is a guardian in a mint-colored uniform." The description of the guardian is "A killjoy wielding a gigantic toothbrush." The guardian carries a gigantic toothbrush. The description of the toothbrush is "Bristles as long as your hand. Firm bristles, too, not those soft ones. The guardian doesn't care about your tender gums."

A thing can be sweet. The butterscotch path, the lollipop, and the gumdrop are sweet.

Carry out showing a sweet thing to the guardian:
    say "The guardian shrieks! You don't understand its language, but from its ululations you understand the idea of decay. There may have been a bit in there about a root canal." instead.

Carry out showing something to the guardian:
    say "The guardian nods approvingly at the unsweetened [noun]." instead.

```
Report eating a sweet thing in the presence of the
guardian:
    say "The guardian looks mournful, but unholsters his
tube of paste and begins applying it to the toothbrush,
as though to say that he really did not want to have to
do this...";
    end the story saying "Everything goes minty" instead.


Report eating something:
    say "You consume [the noun] with gusto." instead.


Test me with "x guardian / x toothbrush / show gumdrop
to guardian / show path to guardian / show tuna to
guardian / look / eat gumdrop".
```

Note that because we only deactivate the carrying requirements rule for showing purposes, the player still takes the gumdrop before eating it.

---

87 ExampleSand

Extend PUT and INSERT handling to cases where multiple objects are intended at once. *WI*

The above example does not quite work when we want the player to be allowed to take multiple objects at once before putting them somewhere: we also need to add a couple of "understand" rules borrowed from many chapters later. While the reasons may not be immediately clear, we include the demonstration here for the sake of thoroughness:

```
"Sand"


Before inserting something which is not carried by the
player into something:
    if the noun is in the second noun, say "Already done."
instead;
    say "(first taking [the noun]) ";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.
```

Before putting something which is not carried by the player on something:
    if the noun is on the second noun, say "Already done." instead;
    say "(first taking [the noun])[line break]";
    silently try taking the noun;
    if the player is not holding the noun, stop the action.

Understand "put [things] in [something]" as inserting it into. Understand "put [things] on [something]" as putting it on.

The Closet is a room.

A lentil is a kind of thing. A black-eyed pea is a kind of thing. The closet contains 3 lentils. The Closet contains 14 black-eyed peas. The round tin is a container in the closet. The round tin contains 17 lentils. The square tin is a container in the Closet. The square tin contains 20 black-eyed peas.

Sorting is a scene. Sorting begins when play begins. Sorting ends when all the lentils are in the round tin and all the black-eyed peas are in the square tin. When Sorting ends, end the story finally.

When play begins: say "Thanks to your cruel stepmother, you're not going anywhere until the lentils and peas are sorted."

Test me with "put peas in square tin / put lentils in round tin".

An effect that occurs only when the player leaves a region entirely.

Suppose that we want to have something happen when the player leaves a region we've defined. "Instead of going from (the region)..." will not suffice for this, because this rule will be invoked every time the player successfully leaves a room within the region, whether or not he is going to a room that is also in the same region.

Instead we need a rule that is a bit more specific, like this:

```
"Veronica"

Neptune is a region.

Tijuana is a room.

High School is north of Tijuana. It is in Neptune.

Detective Offices is west of High School. It is in
Neptune.

The player is in High School.

Instead of going from Neptune to a room which is not in
Neptune:
    say "It's a bad time to leave Neptune."
```

Test me with "s / w / e".

Example Mattress King

Adding extra phrasing to the action to PUSH something in a direction.

By default, when the player pushes something a direction, Inform checks to make sure that the object is pushable between rooms. If not, it blocks the action; if so, it carries out a normal going action with the pushed object taken along.

Also by default, this action produces only a description of the new room that we've traveled into. But suppose we would like to print a short message describing the pushing action first:

```
"Mattress King"

Monica's Bedroom is a room. The Living Room is south
of Monica's Bedroom. Rachel's Bedroom is south of the
Living Room.

After going a direction (called way-pushed) with
something (called the thing-pushed):
    say "You push [the thing-pushed] [way-pushed] to
[the location].";
    continue the action.

The race car bed is an enterable supporter in Monica's
Bedroom. It is pushable between rooms.

Test me with "push bed south".
```

Example **The Second Oldest Problem** *WI*

Adapting the going action so that something special can happen when going from a dark room to another dark room.

Text in this example is drawn from Will Crowther's original 1976 FORTRAN implementation of ADVENTURE, the founding work of the genre, whose source code was rediscovered by Dennis G. Jerz in 2007. Note the capitals: the program ran on an early computer without lower case lettering. They look a little mimsy now, but picture them glowing green on an old-style cathode ray tube monitor in a darkened room late at night.

The problem alluded to is that the player is forbidden to walk between two dark rooms, so that he must always have light to see by from at least one end of any movement. Writing source text to achieve this is tricky to get right in every case, because the determination of light is hard to do. Here we interleave the necessary rules into the existing "going" action, using a new action variable to record the number of ends which are dark as experienced by the player, which might be 0, 1 or 2:

```
"THE SECOND OLDEST PROBLEM"


The going action has a number called the dark terminus
count.
Setting action variables for going:
    now the dark terminus count is 0;
    if in darkness, increment the dark terminus count.
The last carry out going rule:
    if in darkness, increment the dark terminus count;
    if the dark terminus count is 2, end the story saying
"YOU FELL INTO A PIT AND BROKE EVERY BONE IN
YOUR BODY!" instead.
```

And now three early rooms to try this out.

```
COBBLE CRAWL is a room. "YOU ARE CRAWLING
OVER COBBLES IN A LOW PASSAGE. THERE IS A
DIM LIGHT AT THE EAST END OF THE PASSAGE."
```

DEBRIS ROOM is west of COBBLE CRAWL. "YOU ARE IN A DEBRIS ROOM, FILLED WITH STUFF WASHED IN FROM THE SURFACE. A LOW WIDE PASSAGE WITH COBBLES BECOMES PLUGGED WITH MUD AND DEBRIS HERE,BUT AN AWKWARD CANYON LEADS UPWARD AND WEST."

AWKWARD CANYON is west of DEBRIS ROOM. "YOU ARE IN AN AWKWARD SLOPING EAST/WEST CANYON."

DEBRIS ROOM and AWKWARD CANYON are dark.

Rule for printing the name of a dark room: say "DARKNESS" instead.
Rule for printing the description of a dark room: say "IT IS NOW PITCH BLACK. IF YOU PROCEED YOU WILL LIKELY FALL INTO A PIT." instead.

Test me with "w / e / w / w".

This is only the second oldest problem in the IF literature: the earliest puzzle is unlocking the steel grate which bars entrance to the cave.

ExampleMisadventure

A going by name command which does respect movement rules, and accepts names of rooms as commands.

The original Adventure allowed the player to type the names of rooms in order to move to them, and it is now not too difficult for us to do the same. Adventure restricted this option to adjacent rooms, but we might want to be a bit more flexible, so we will accept any room:

"Misadventure"

```
Plover Room is a room. "You're in a small chamber lit by
an eerie green light. An extremely narrow tunnel exits to
the west. A dark corridor leads northeast."


The Dark Corridor is northeast of Plover Room. Plover
Room is south of the Dark Corridor. The printed name of
the Dark Corridor is "Dark Room". The description of the
Dark Corridor is "You're in the dark-room. A corridor
leading south is the only exit."


The Alcove is west of Plover Room. "You are in an
alcove. A small northwest path seems to widen after a
short distance. An extremely tight tunnel leads east. It
looks like a very tight squeeze. An eerie light can be
seen at the other end."


Northwest of the Alcove is the Misty Cavern. The
description of Misty Cavern is "You are following a wide
path around the outer edge of a large cavern. Far
below, through a heavy white mist, strange splashing
noises can be heard. The mist rises up through a fissure
in the ceiling. The path exits to the south and west."
West of Misty Cavern is the Alcove.


Understand "[any room]" as going by name. Understand
"go to [any room]" as going by name.


Going by name is an action applying to one thing.
```

We should reject movement to the player's current location, or to anywhere he hasn't been and can't see:

```
Check going by name:
    if the noun is the location, say "You're already in [the
```

```
location]." instead;
    if the noun is not adjacent and the noun is unvisited,
say "That noun did not make sense in this context."
instead.
```

The assumption here is that the player does know the names of the rooms adjacent to his current location, even if he hasn't been there yet.

Now for the travel itself. The simplest way to ensure that the usual movement rules will still apply is to convert GO BY NAME into a GO action, and here the best route comes to our aid:

```
Carry out going by name:
    let aim be the best route from the location to the
noun, using doors;
    if aim is not a direction, say "You can't think how to
get there from here." instead;
    say "(heading [aim])[command clarification break]";
    try going aim;
    if the location is not the noun, say "You'll have to stop
here."
```

This will allow the player to travel toward rooms he has already visited even if they are several moves away.

Finally, so that the player can also use the names of doors as commands:

```
Understand "[door]" as entering.
```

And in keeping with the original, we might add to our scenario a rule or two about restrictions on movement, just to test that it's all working right:

```
The player carries a plover egg and a platinum pyramid.
The description of the egg is "Plover's eggs, by the way,
are quite large." The printed name of the egg is
"emerald the size of a plover's egg". Understand
"emerald" as the egg. The description of the pyramid is
"The platinum pyramid is 8 inches on a side!"


Instead of going to the Plover Room from the Alcove
```

```
when the player carries something which is not the
plover egg:
    say "Something you're carrying won't fit through the
tunnel with you. You'd best take inventory and drop
something."
```

```
Test me with "go to misty cavern / go to dark corridor /
go to plover room / go to alcove / go to dark corridor /
drop pyramid / go to dark corridor / g / go to alcove / g /
go to misty cavern".
```

---

372 Example Minimal Movement *WI*

Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

Sometimes it would be nice to respond a little more sensitively to a vague command such as "leave" -- converting it, perhaps, to a "go out" command.

```
"Minimal Movement"
```

```
The Doll-like House is a room. The Postage-Stamp-
Sized Garden is outside from the House.
```

```
Rule for supplying a missing noun while going:
    now noun is outside.
```

This particular situation is very slightly complicated by the existing rules about vague movement, but fortunately we can easily turn those off.

```
The block vaguely going rule is not listed in the for
supplying a missing noun rules.
```

Test me with "go".

Example **Saint Eligius** *WI*

Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

A not-infrequent desire in IF is to provide a few lines of comment when the player first enters a new room, after the objects are described but before anything else (such as an every turn rule) can fire. The cleanest, most systematic solution is to add a rule to the carry out looking rulebook, so:

"Saint Eligius"

The first look rule is listed after the room description paragraphs about objects rule in the carry out looking rules. A room can be commented or uncommented. A room is usually uncommented.

This is the first look rule:
    if the location is uncommented, carry out the gawking at activity with the location.

Gawking at something is an activity.

Rule for gawking at the Diamond Market:
    say "Your throat closes and your eyes begin to sting. You have long disdained pomp and luxury, and railed against the oppression that brings such wealth to some men at the cost of the lives of others; you were not prepared for the magnificence."

After gawking at a room (called the target): now the target is commented.

And now the scene itself:

The Cobbled Alley is a room. "The Alley has never been made into a proper street: the buildings on either side are simply too important to tear down. For all that, there isn't much sign of the magnificence nearby. The entrance you seek is set below street level, four grimy steps down to a half-basement."

After going to Diamond Market:
    say "You descend the steps quickly and step into the small foyer, allowing yourself to be searched for weapons, before going on to...";
    continue the action.

Diamond Market is down from Cobbled Alley. "The roof is vaulted and painted in allegorical images representing Plenty, the Riches of the Earth, and Saint Eligius, patron of goldsmiths and jewelers.

Under their watchful eye, dozens of men in sober black robes sit; and on the tables before them are rubies, emeralds, sapphires from oversea, but most of all diamonds, both raw and cut."

The burly guard is a man in Diamond Market. "A burly guard patrols quite close to you, but even he is more sumptuously dressed than the average burly guard, and his buttons shine."

Test me with "d / look".

Example **Up and Up**

Adding a short message as the player approaches a room, before the room description itself appears.

Sometimes when a player moves from one room to another, we want to imply that a considerable amount of time elapses, or that something interesting occurs on the way. In that case, we might want to print more than just the room description itself. Here is how we might define a couple of rooms that are far apart:

```
"Up and Up"

The Plain of the Skull is below the Endless Tower. The
description of the Plain of the Skull is "A vast and
trackless plain, enlivened only by the bones of those
who have previously tried and failed to cross. Above you
is the Endless Tower, which rises half-way to the
moon."

The description of the Endless Tower is "From up here
the Plain of the Skull seems only a small bald patch: the
world is round and most of it is covered with trees. Far
off to the southwest is a shimmering surface that might
be water; but there are no signs of cities or civilizations,
only the lizard-skeletons."
```

And now we borrow from the instructions on Actions to create our actual message. "Before..." introduces a rule that occurs when the player tries to do something; in this case, we will make a Before rule for going to the tower.

```
Before going to the Endless Tower:
    say "You climb... and climb... and climb... The sun
sets. The moon rises. The wind begins to blow. You
continue to climb..."
```

The player carries a bit of harness. The description of the harness is "A strip of worked leather and a loop of metal, scavenged from one of the skeletons on the plain. Without it, you might think your entire quest was in vain."

Test me with "look / up".

Hiking Mount Rainier, with attention to which locations are higher and which lower than the present location.

Suppose we have a landscape with a great deal of up and down variation, where GO UP and GO DOWN will be significant almost everywhere, and specifying them all individually a tremendous pain:

"Wonderland"

An altitude is a kind of value. 1000 feet specifies an altitude. A room has an altitude.

Definition: a room is low if its altitude is 3000 feet or less. Definition: a room is high if its altitude is 5000 feet or more.

```
Instead of going down:
    if an adjacent room is lower than the location:
        let the valley be the lowest adjacent room;
        let the way be the best route from the location to
the valley;
        say "(that is, [way])[paragraph break]";
```

```
        try going the way;
    otherwise:
        say "You're in a local valley: there's no down from
here."


Instead of going up:
    if an adjacent room is higher than the location:
        let the peak be the highest adjacent room;
        let the way be the best route from the location to
the peak;
        say "(that is, [way])[paragraph break]";
        try going the way;
    otherwise:
        say "You're on a local peak."


Paradise is a room. Paradise has altitude 5400 feet. "A
handsome parking lot, a picnic ground, and the Henry
M. Jackson Memorial Visitor Center. The latter offers,
for serious climbers, a hot shower; for nature
enthusiasts, an interpretive museum; and for car-
trippers, a gift shop selling canned slugs. All of which is
a largely unsuccessful distraction from the peak of Mt.
Rainier beyond."


Cougar Rock is southwest of Paradise. The altitude of
Cougar Rock is 3180 feet. "Numerous individual
campsites and (on the road inventively labeled 'F') a
handful of larger campgrounds suitable for church
groups and family reunions."


Longmire is southwest of Cougar Rock. It has altitude
2760 feet. "A tiny town: it has to offer a few groceries, a
post office, and a lodge for people who do not care to
camp, all built in a rustic Park Service way."


Panorama Point is north of Paradise. It has altitude
6800 feet. Camp Muir is north of Panorama Point. It has
altitude 10188 feet. Columbia Crest is northwest of
Camp Muir. It has altitude 14410 feet. St Andrews Rock
```

```
is west of Columbia Crest. It has altitude 10992 feet.
Camp Schuman is northeast of Columbia Crest. It has
altitude 9510 feet.
```

Since Mount Rainier National Park runs to over 235,000 acres, we will omit the rest of the locations, but it does seem fair to give a little more credit to anyone who makes the summit:

```
Instead of going up in the highest room:
    say "You're standing at the summit of Mt. Rainier, the
highest point in the state of Washington. There is no
up."

Test me with "up / up / up / down / down / up / up".
```

The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling all openable doors on the way.

The foregoing example moves the player one location towards his destination, and requires that rooms have been visited before. But suppose we wanted to be a bit more lenient about movement, and let the player make as many steps as necessary per turn. We will also show consideration about doors, using the "Locksmith" extension supplied with Inform. (Now every time the code attempts opening a door, unlocking rules will also be invoked.)

```
"Safari Guide"

Include Locksmith by Emily Short.

The Monkey House is a room. The African Grasslands
Exhibit is north of the Monkey House. The bird door is
```

```
north of the African Grasslands Exhibit and south of the
Aviary. The Ostrich Enclosure is west of the Aviary. The
bird door is a door. It is closed, lockable, and locked.
The silver key is a passkey. It unlocks the bird door. The
player carries the silver key.



Understand "go to [any room]" as going by name.
Understand "[any room]" as going by name. Understand
"[door]" as entering.



Going by name is an action applying to one thing.



Check going by name:
    if the noun is the location, say "You're already in [the
location]." instead.



Carry out going by name:
    while the player is not in the noun:
        let heading be the best route from the location to
the noun, using even locked doors;
        if heading is not a direction, say "You can't think
how to get there from here." instead;
        let destination be the room heading from the
location;
        say "(heading [heading])[command clarification
break]";
        try going heading;
        if the player is not in the destination, rule fails.



Test me with "go to aviary / go to ostrich enclosure /
african grasslands".
```

Notice that we continue the movement until one of two things happens: either the player reaches the room that is his destination, or the going attempt doesn't work. In the latter case we stop the action in order to avoid hanging the game up in a loop. This event might occur when the player runs into a locked door, for instance.

ExampleBumping into Walls

Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

```
"Bumping into Walls"
```

First we add an instruction to determine which ways lead to other rooms.

```
Definition: a direction (called thataway) is viable if the
room thataway from the location is a room.
```

Now we build in the instruction for what Inform should say if the player tries to head in a direction that leads nowhere:

```
Instead of going nowhere:
    let count of exits be the number of viable directions;
    if the count of exits is 0, say "You appear to be
trapped in here." instead;
    if the count of exits is 1, say "From here, the only way
out is [list of viable directions].";
    otherwise say "From here, the viable exits are [list of
viable directions]."
```

There is no theoretical reason why we have to define "count of exits" here: we could, if we wanted, just say "if the number of viable directions is 0", "if the number of viable directions is 1", and so on. However, each calculation of a "viable direction" takes a bit of computing power, so there is some slight savings in not requiring the game to count viable directions more than once in this routine.

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
```

```
southwest of Dome, west of South Chapel, and south of
Western End.


The Crypt is below the dome.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "u / n / n / e / n / s / u / open door / e / n".
```

ExamplePolarity *WI*

A "go back" command that keeps track of the direction from which the player came, and sends him back.

The main trick of this is always to record where the player has gone when he has just moved.

```
"Polarity"


The former location is a room that varies.
```

Here we record where the player has been before moving him; by calling this the "first carry out going rule", we make sure that this rule is followed during the going action before any other pieces of the movement occur. For more detail, see the chapters on advanced actions and on rules.

```
First carry out going rule:
    now the former location is the location.
```

```
Understand "go back" as retreating. Understand "back"
or "return" or "retreat" as retreating.


Retreating is an action applying to nothing.


Carry out retreating:
    let way be the best route from the location to the
former location, using doors;
    if way is a direction, try going way;
    otherwise say "You can't see an open way back."
```

And to deal with the case where the player has not yet moved:

```
When play begins: now the former location is the Dome.


Instead of retreating when the former location is the
location: say "You haven't gone anywhere yet."


Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "back / n / go back / e / open door / go
through door / go back".
```

ExampleProvenance Unknown

Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

Suppose we have a series of items that might be stacked on top of one another -- say a heavy television on a rolling cart, and we want the player to be able to move the cart with PUSH TELEVISION EAST just as well as with PUSH CART EAST.

This takes a little redirection, using a setting action variables rule. This is not a kind of rule we've encountered yet, and in fact we won't meet it until the Advanced Actions chapter; it is included here for the convenience of authors who want to modify the effect of pushing without reading that far ahead:

```
"Provenance Unknown"

Setting action variables for pushing something to:
    if the noun is enclosed by a pushable between rooms
thing (called the pushed item) which is in the location:
        now the noun is the pushed item instead.
```

This rule says that any time we push an object that is on top of a stack of pushable objects, we should transfer the action to the item at the bottom of the stack.

The rest is merely a test case.

```
The heavy golden idol is on a roller board. The roller
board is on a hovercraft.

The hovercraft, the tea trolley, and the skateboard are
pushable between rooms.
```

The hovercraft is in Zeta Proximan Dig Field.

Zeta Proximan Dig Field is a room. "During the day, the field is massed with sweating native workers, overseers, and officials from central command. Now the spades, trowels, brushes, metal detectors, ground probes, plumb lines, and sighting tripods have been laid aside.

All that remains are the trenches and the fine grey dust that blows slowly across them; the moonlight; and the just-emerging outlines of an ancient and alien wall."

The Hover-Road is west of the Dig Field. "A long road hastily laid down, stretching east to west, from the dig site toward the safety of the city."

When play begins:
    say "You have, at last, loaded your illicit cargo without setting off any of the many and sensitive alarms set here; now it remains only to sneak out of the area, under the light of Zeta Proxima's lone green moon."

Test me with "push idol west / look / push roller board east / look".

Replacing the message the player receives when attempting to push something that isn't pushable, and also to remove the restriction that objects cannot be pushed up or down.

There are two aspects of Inform's handling of pushable objects that are particularly prime for modification. One is that we may want to change the language used to refuse the pushing of unpushable objects.

Second, Inform by default assumes that it is impossible to push objects in

up or down directions. This makes lots of sense if the player is trying to push a wheelbarrow up a ladder; it makes less sense if instead we're pushing a ball up a slope.

We solve both problems with some syntax borrowed from the chapter on rulebooks: in the first case, we replace the old rule with a new one with more friendly phrasing; in the second, we remove the rule entirely. More about how to do this is described in the rulebooks chapter; and in general we can find out what rules contribute to any given action by looking at the Actions index. In this case, the action is "pushing it to", which has its own set of prerequisites (called check rules) that make sure the object can safely be pushed, before turning processing over to the going action.

```
"Zorb"

Section 1 - Procedure

The new can't push unpushable things rule is listed
instead of the can't push unpushable things rule in the
check pushing it to rules.

This is the new can't push unpushable things rule:
    if the noun is not pushable between rooms:
        say "[The noun] [are] not amenable to being
pushed from place to place." instead.

The can't push vertically rule is not listed in any
rulebook.
```

And now to provide a scenario where the player can push something up and down a hillside. Most of the rest of the example is there for local color and to provide a way to demonstrate these rule adjustments:

```
Section 2 - Scenario
```

The Steep Hill is a room. The Crest is above Steep Hill. The Valley is below Steep Hill.

The flat rock is a fixed in place thing in the Steep Hill.

The Zorb is a transparent open enterable container in the Steep Hill. "[if the player props the Zorb]The Zorb rests here, kept from further rolling by your support[otherwise]The Zorb is here[end if].". It is pushable between rooms. The description of the Zorb is "A giant plastic inflatable ball, like a hamster ball for humans[if someone is in the Zorb]. Inside [is-are list of people in the Zorb][end if]."

Lucy is a woman in the Zorb.

Carry out going with the Zorb when the Zorb contains Lucy:
    say "Lucy whoops delightedly as she rides along in the Zorb."

Every turn when the Zorb is not in the Valley and the player does not prop the Zorb:
    let next room be the room down from the location of the Zorb;
    if the player is not in the Zorb and the player can see the Zorb:
        say "The Zorb succumbs to gravity and rolls down toward [the next room].";
    move the Zorb to the next room;
    if the player is in the Zorb:
        say "The Zorb rolls you down the hill!";
        try looking;
    otherwise if the player can see the Zorb:
        say "The Zorb rolls ponderously but inevitably into the vicinity.";

Propping relates one person to one thing. The verb to prop means the propping relation.

Carry out going with the Zorb:
    now the player props the Zorb.

Before doing something when the action requires a touchable noun:
    if the noun is not the Zorb, now the player does not prop the Zorb.

Check waving hands when the player is propping something (called casualty):
    try the player releasing the casualty.

Carry out entering the Zorb:
    now the player does not prop the Zorb.

Understand "let go of [something]" or "let [something] go" or "release [something]" or "free [something]" as releasing. Releasing is an action applying to one thing.

Check releasing:
    if the player carries the noun:
        try dropping the noun instead.

Check releasing:
    if the player does not prop the noun:
        say "You are not supporting [the noun]." instead.

Carry out releasing:
    now the player does not prop the noun.

```
Report releasing:
    say "You let go of [the noun]."
```

```
Test me with "d / push zorb up / look / push zorb up /
wave / d / d / push zorb up / release zorb / d / push zorb
up / touch rock / push the flat rock south".
```

ExampleOwen's Law

OUT always means "move to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

Suppose we want the game to interpret "GO OUT" as "move towards an outdoors room, or towards a room with more exits than the current room", while "GO IN" means "move toward a room with fewer exits, or towards an indoors room". Thus going in repeatedly within a building would lead towards dead-ends, while going out repeatedly would lead towards the center of the building and then towards an exit to the outside world.

We start by encoding these rules as definitions:

```
"Owen's Law"
```

```
A room can be indoors or outdoors. A room is usually
indoors.
```

```
Definition: a room is outward:
    if it is not adjacent, no;
    if it is indoors and the location is outdoors, no;
    if it is outdoors and the location is indoors, yes;
    if the number of rooms adjacent to it is greater than
the number of rooms adjacent to the location, yes;
    otherwise no.
```

```
Definition: a room is inward:
    if it is not adjacent, no;
    if it is outdoors and the location is indoors, no;
    if it is indoors and the location is outdoors, yes;
    if the number of rooms adjacent to it is less than the
number of rooms adjacent to the location, yes;
    otherwise no.



Instead of going nowhere when the noun is outside: try
exiting.



Instead of exiting when the player is in a room:
    if at least one room is outward:
        let the destination be a random outward room;
        let the way be the best route from the location to
the destination;
        say "(that is, [way])[command clarification break]";
        try going the way instead;
    otherwise:
        say "It's not entirely obvious which way you mean.
";
        carry out the listing available exits activity.



Instead of going inside when the room inside from the
location is not a room and at least one room is inward:
    if more than one room is inward:
        carry out the listing available exits activity;
    otherwise:
        let the destination be a random inward room;
        let the way be the best route from the location to
the destination;
        say "(that is, [way])[command clarification break]";
        try going the way instead.



Instead of going nowhere:
    carry out the listing available exits activity.
```

This "listing available exits" is a refinement borrowed from a future chapter, which allows us to specify special listing and printing rules:

Listing available exits is an activity.

Rule for listing available exits:
    if going inside and an adjacent room is inward:
        say "From here 'in' could reasonably mean [a list of adjacent inward rooms].";
        rule succeeds;
    if exiting and an adjacent room is outward:
        say "From here 'out' could reasonably mean [a list of outward adjacent rooms].";
        rule succeeds;
    say "From here you can go [a list of adjacent rooms]."

Before printing the name of a room (called the target) while listing available exits:
    let aim be the best route from the location to the target;
    say "[aim] to the ".

Rule for printing the name of an unvisited room which is not the location:
    say "unknown location".

Dune is an outdoors room. "Hundreds of feet of dune stretch west to the beach, crisscrossed with dune-buggy tracks and the footprints of birds. To the east is a low-lying, boxy concrete installation."

Ocean Shores Military Installation is east of the Dune. It is an outdoors room. "The World War II emplacements, built in case of Japanese invasion, have never been destroyed, though with all the weapons and furnishings gone it is difficult to make much sense of the original structure. A doorway leads west into concrete-lined darkness; a rusty but reliable ladder ascends to a walkway overlooking the sea."

Walkway is above Ocean Shores Military Installation. "From here you have a long view of the dunes and the Pacific Ocean, complete with the rotting hull of a long-stranded vessel."

Dark Echoing Room is inside from Ocean Shores Military Installation. Dank Dripping Room is east of Dark Echoing Room. Narrow Room is south of Dark Echoing Room. Small Sealed Chamber is north of Dark Echoing Room. Room Smelling of Animal Urine is north of Dank Dripping Room. The description of a room is usually "It is dark in here, and feels unsafe."

Test me with "e / u / d / in / s / out / n / out / e / in / out / out / out".

Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

We could now re-write the check rules so that any time someone (the player or someone else) tries to pick up a container which he is in, he will first get out:

"GET AXE"

This is the clever can't take what you're inside rule:
    if the person asked is in the noun, try the person asked exiting;
    if the person asked is in the noun, rule fails.

The clever can't take what you're inside rule is listed instead of the can't take what you're inside rule in the check taking rules.

Attic is a room. The unused coffin is in the Attic. The coffin is enterable and openable and open. Raskolnikov is a man in the coffin.

Persuasion rule for asking Raskolnikov to try doing something:
    persuasion succeeds.

Test me with "raskolnikov, get coffin".

ExampleAnchorite

By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.

With GET DOWN, we can replace the whole command, which will not interfere with the normal function of the TAKE verb, or allow the player to attempt to GET any other directions:

"Anchorite"

The Solitary Place is a room. "A glittering, shimmering desert without either locusts or honey." The pillar is an enterable supporter in the Solitary Place. "The broken pillar is short enough to climb and sit on." The description of the pillar is "Once it was a monument: a long frieze of battles and lion-hunts spirals up the side, in honor of an earthly king." The player is on the pillar.

```
Understand "get down" as exiting.
```

This doesn't cover the case where the player just types "DOWN", and we don't want to preempt the normal operation of the GO action here. So instead of writing a new understand instruction, we might catch this one at the action-processing level:

```
Instead of going down when the player is on a
supporter:
    try exiting.


Test me with "down / enter pillar / get down / down / get
down".
```

Commands to allow the player to lie down in three different ways.

To set the scene, and make new actions to provide for two of these ways:

```
"Lies"


The Laundry is a room. "An old Limehouse haunt, the
Chinese laundry used by the down-trodden wives of the
Tong of the Black Scorpion." The vast marble sink is
here. "There is nothing obviously oriental about the vast
marble sink, which is large enough to lie down inside. A
wooden-rack floor, equipped for easy drainage, turns
out also to be equipped for snagging the shoes of
passers-by." The sink is an enterable container, fixed in
place.
```

```
Lying down is an action applying to nothing. Report lying
down: say "You lie down for a while in the middle of the
Laundry, wondering about the point of existence, then
get up again."


Lying near is an action applying to one thing. Report
lying near: say "You lie down next to [the noun] for a
while, mumbling to yourself."


Instead of lying near the sink, say "Lying down close to
the cool butcher's marble slabs of the sink, your
attention is caught by the sight of coolie shoes through a
floor-level grille for ventilation. The game is afoot!"
```

So far, so good. Now for the grammar, where we create two new tokens: one for each of two groups of alternative prepositions.

```
Understand "beneath/under/by/near/beside/alongside/
against" or "next to" or "in front of" as "[beside]".


Understand "on/in/inside" or "on top of" as "[within]".


Understand "lie down" as lying down.


Understand "lie down [within] [something]" as entering.


Understand "lie [beside] [something]" or "lie down
[beside] [something]" as lying near.


Test me with "lie down / lie down on top of the sink / get
out / lie down inside the sink / get out / lie down in front
of the sink".
```

Example **Nine AM Appointment**

WI

A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

If there's some reason the player needs to be at a specific place and time, we might want to allow him to wait a number of minutes at once.

```
"Nine AM Appointment"

Waiting more is an action applying to one number.

Understand "wait [a time period]" or "wait for [a time
period]" or "wait for a/an [a time period]" or "wait a/an [a
time period]" as waiting more.

Carry out waiting more:
    let the target time be the time of day plus the time
understood;
    decrease the target time by one minute;
    while the time of day is not the target time:
        follow the turn sequence rules.
```

The one nuance here is that after our wait command occurs, the turn sequence rules will occur one more time. So we need to subtract one minute from the parsed time to make the turn end on the desired number of minutes.

```
Report waiting more:
    say "It is now [time of day + 1 minute]."
```

And if we want to ensure that the player doesn't (accidentally or intentionally) put the interpreter through a really long loop, we could put an upper limit on his patience:

Check waiting more:
    if the time understood is greater than one hour, say "You really haven't got that kind of patience." instead.

The Specialist's Office is a room. The secretary is a woman in the Office. Instead of asking the secretary about "[appointment]", say "'Hang on just five more minutes,' she says, in a distracted manner."

Understand "appointment" or "specialist" or "doctor" as "[appointment]".

At 9:45 AM: say "At [the time of day in words], secretary glances at you and gives a reassuring smile."

Test me with "ask secretary about appointment / wait five minutes / g / g / wait 61 minutes / wait for half an hour / wait for a quarter of an hour / wait for an hour".

Example **Change of Basis**

Implementing sleeping and wakeful states.

Suppose we want to allow the player to go to sleep some of the time:

"Change of Basis"

A person is either awake or asleep. A person is usually awake.

The important thing to note here is that it does not work to say "the player is either asleep or awake". This is because the player is not necessarily one specific person or thing during the game: the identity of the player can be changed, as we will see later.

So if we want to make rules about the properties of the player, we should attach these rules to the "person" kind.

```
Linear Algebra Class is a room. "The blackboard is
covered with square arrangements of numbers. These
are supposed to convey something to you, but mostly
you're finding them soporific."
```

Now a few rules about changing from one state to the other:

```
Instead of sleeping: now the player is asleep; say "You
drop off."


Instead of doing something other than waking up,
waiting or sleeping when the player is asleep:
    say "Ssh! You're sleeping!"


Instead of sleeping when the player is asleep:
    say "Zzzz."


Instead of waking up when the player is asleep:
    now the player is awake;
    say "You come to suddenly, wiping drool from your
lips."


Instead of doing something other than looking or
sleeping when the player is awake:
    say "You'd really rather just sleep through this."
```

Test me with "wake up / sleep / look / z / sleep / wake up / look".

Example **Delayed Gratification**

A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

```
"Delayed Gratification"

Hanging around until is an action applying to one time.

Check hanging around until:
    if the time of day is the time understood, say "It is
[time understood] now!" instead;
    if the time of day is after the time understood, say "It
is too late for that now." instead.

Carry out hanging around until:
    while the time of day is before the time understood:
        follow the turn sequence rules.

Report hanging around until:
    say "You yawn until [time understood]."

Understand "wait until [time]" as hanging around until.

The Empty Field is a room. "It's an ordinary empty field.
Nothing to see here at all-- yet. Wait until 11:45 PM,
though."
```

```
At 11:45 PM:
    say "Suddenly the air is filled with light and the
sounds of an approaching band. Over the crest of the
hill comes a parade of singing, stomping, hooting
people: and not just people, but dogs, horses,
elephants, giraffes... There are banners, and candles,
and a flag that glows eerie-green in the dark; there is a
float shaped like an enormous turtle, its shell covered
with winking green lights; there is an old man dressed
as a skeleton, carried in a litter, his neck garlanded with
dried chiles. There are small girls throwing rose petals
from a basket, and grown women half-naked carrying
the emblems of Bacchic revelry, and two little boys each
with a silver basin of clear water. All these go by in
procession, and you join on at the end.";
    end the story finally.


Test me with "look / z / z / wait until 11:45 PM".
```

ExampleXYZZY

Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

We have seen before how to define a new action from scratch, but we may want to review here, using a simple command that requires no objects.

```
"XYZZY"


Understand "xyzzy" or "say xyzzy" or "cast xyzzy" as
casting xyzzy.


Casting xyzzy is an action applying to nothing.
```

Check casting xyzzy:
    if the player does not wear the amulet of elocution, say "You are unable to articulate the second 'z' separately from the first, and the spell fails in a disdainful puff. Must be Parisian magic." instead;
    if the player has the plate, say "The plate of cheeses twitches uncomfortably, aware that it should be doing something, but not sure what." instead.

Carry out casting xyzzy:
    move the plate to the player.

Report casting xyzzy:
    say "Under the influence of the Amulet of Elocution, you pronounce this as Xhi-zee. And lo, from nowhere, a [plate] appears!"

The amulet of elocution is a wearable thing. It is carried by the player. The description is "A heavy gold ring on a chain. If heated in an ordinary house fire, it glows with the words, 'Moses Supposes His Toeses Are Roses.'"

The plate is a portable supporter. On the plate is a very ripe ooze. Instead of smelling the ooze, say "It smells like socks. This is going to be wonderful." The ooze is edible. The printed name of the plate is "plate[if the plate supports the ooze] of cheese[end if]". The description of the ooze is "Definitely genuinely cheese." Understand "cheese" as the ooze.

Instead of eating the ooze: now the ooze is nowhere; say "You are transported..."; move the player to Paradise.

The Cheez Factory is a room. "All around you are squares of pressed orange polymer, or possibly cheez. Your only hope is the magic word your uncle taught you: XYZZY." The squares of pressed orange polymer are scenery in the Factory. The description is "You see nothing special about the squares of pressed orange polymer. Nothing special at all." Understand "square" or "cheez" as the squares.

Paradise is a room. The description is "Well, it might just be one of the posh upper rings of purgatory, if you're entirely honest with yourself."

Test me with "x squares / x amulet / x cheese / xyzzy / wear amulet / xyzzy / x ooze / smell ooze / eat ooze".

XYZZY is a magic word from the original Adventure, and many other games respond to it with some sort of amusing message.

ExampleFine Laid

Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

Sometimes it is useful to direct all -- or almost all -- actions from one object to another. For the sake of argument, say we have a sheet of paper with writing on it, and (because we're very meticulous) we want to let the player examine the writing and get a customized response, different from when he just examines the sheet of paper. But for all other purposes -- say, TAKE or TASTE -- we want the two objects to be treated as one.

Here, we approach the problem by changing the noun and/or the second noun of the current action, then issuing a new command to "try the current action". Because we've changed the noun and second noun, the "current action" at this point is different from the one generated originally by the player's command.

"Fine Laid"

```
High Street Stationer is a room.


The sheet of paper is a thing in High Street Stationer.
The writing is part of the sheet of paper.


The description of the sheet of paper is "A beautiful
sheet of heavy cream paper." The description of the
writing is "Delicate and spidery."


Instead of tasting the sheet of paper, say "You might
need more fiber in your diet, but this isn't the way.".


Before doing something other than examining when the
current action involves the writing:
    if the writing is the noun, now the noun is the sheet of
paper;
    if the writing is the second noun, now the second
noun is the sheet of paper;
    try the current action instead.


Test me with "examine sheet of paper / examine writing
/ get writing / taste writing".
```

ExampleLucy

Redirecting a question about one topic to ask about another.

Occasionally we will want to replace the player's question topic with another of our own devising. We can do this in the simplest possible case like so:

```
"Lucy"

The International Boardgame Championship is a room.
Lucy is a woman in the Championship.

Instead of asking Lucy about "checkers":
    try asking Lucy about "games".

Instead of asking Lucy about "games",
    say "'I don't like games,' she sniffs."

Test me with "ask lucy about checkers / ask lucy about
games".
```

Note that this syntax did not work in older versions of Inform; it is now safe.

Example **Cactus Will Outlive Us All** *WI*

For every character besides the player, there is an action that will cause that character to wither right up and die.

```
"Cactus Will Outlive Us All"

Death Valley is a room. Luckless Luke and Dead-Eye
Pete are men in the Valley. A cactus is in the Valley.
Persuasion rule: persuasion succeeds.

A person has an action called death knell. The death
knell of Luckless Luke is pulling the cactus. The death
```

```
knell of Dead-Eye Pete is Luke trying dropping the
cactus.



Before an actor doing something:
    repeat with the victim running through people in the
location:
        let the DK be the death knell of the victim;
        if the DK is not waiting and the current action is the
DK:
            say "It looks as if [the DK] was the death knell
for [the victim], who looks startled, then nonexistent.";
            now the victim is nowhere.
```

If we leave it at that, then pulling the cactus will kill Luckless Luke but then say "Nothing obvious happens.", which seems like a bit of an anti-climax. So we add a special case response for that one:

```
After pulling the cactus when Luckless Luke was in the
location:
    say "That's a real shame."



Test me with "get cactus / drop cactus / luke, get cactus
/ luke, drop cactus / pull cactus / look".
```

ExampleLeopard-skin

A maze that the player can escape if he performs an exact sequence of actions.

Suppose (as in Infocom's Leather Goddesses of Phobos) that we have a maze that the player can escape only by performing the correct sequence of actions in the correct order. One way to do this would be to keep a list of the player's most recent actions, and see whether these match up with the combination we have established as the maze's solution.

For instance:

"Leopard-skin"

The Fur-Lined Maze is a room. "This seemingly endless sequence of rooms is decorated in a tasteful selection of exotic furs and gilded fixtures."

Clapping is an action applying to nothing. Understand "clap" as clapping.
Kweepaing is an action applying to nothing. Understand "kweepa" as kweepaing.

Carry out clapping:
    say "You clap."

Carry out kweepaing:
    say "You holler 'KWEEPA!' triumphantly."

The maze-sequence is a list of stored actions that varies.

When play begins:
    add jumping to the maze-sequence;
    add clapping to the maze-sequence;
    add kweepaing to the maze-sequence.

The attempted-sequence is a list of stored actions that varies.

Every turn when the player is in the Fur-Lined Maze:
    truncate the attempted-sequence to the last two entries;
    add the current action to the attempted-sequence;

```
    if the attempted-sequence is the maze-sequence:
        say "That does it! You are instantly transported
from the maze!";
        end the story finally.


Test me with "hop / clap / clap / hop / kweepa / hop /
clap / kweepa".
```

Example I Didn't Come All The Way From Great Portland Street *WI*

In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

There is very little to this, in fact. The tricky rule to enforce is Repetition: the player is forbidden to repeat any previously tried action. We keep track of this by keeping a set of past actions, which for want of a better term is called the "tally". All we need to do is:

```
if the current action is listed in the tally, challenge for
"Repetition of [the current action]!";
otherwise add the current action to the tally.
```

Note that the tally can never contain duplicates, and that when, at the end of the round, we print it out, we sort it first - this makes a more natural-looking sentence. (Sorting a list of actions uses the natural order for actions: compare the sequence on the Actions page of the Index.) The full text, then, is:

```
"I Didn't Come All The Way From Great Portland Street"


The Paris Theatre is a room. An instrument is a kind of
thing. The violin, the tuba, the xylophone and the
triangle are instruments. The violin is inside the case.
The tuba, the xylophone, the radish, the case, the bust
of Nicholas Parsons, the purple felt hat and the triangle
```

are in the Paris Theatre.

The Round is a scene. The Round begins when play begins. The Round ends when the turn count is 10.

The tally is a list of stored actions that varies.

When the Round begins:
    say "'And the subject on the card is... musical instruments. Will you carry out for us something to do with that, please, for ten turns starting - now!'"

When the Round ends:
    sort the tally;
    say "Phweeep![paragraph break]'So, when the whistle goes ten turns are up, you get a point for acting when the whistle blows, and in that round you entertained us by [the tally], and you also get a bonus point for keeping going until the whistle went.'";
    end the story finally.

To challenge for (infraction - text):
    say "Bzzzzt! 'And [one of]Clement Freud[or]Derek Nimmo[or]Kenneth Williams[or]Peter Jones[at random] has challenged.'[paragraph break]'[infraction]'[paragraph break]'Well, as it's your first time playing the game, and the audience was enjoying your contribution so much, I will disallow the challenge, you have [10 minus the turn count] turn[s] left on musical instruments, starting... now!"

Before doing something:
    if the current action is listed in the tally, challenge for "Repetition of [the current action]!" instead;
    otherwise add the current action to the tally;
    if waiting, challenge for "Hesitation!" instead;

```
    if not looking and not waiting and the noun is not an
instrument and the second noun is not an instrument,
challenge for "Deviation!" instead.


Test me with "look / wait / examine bust / take tuba / get
triangle / hit xylophone / get tuba / examine tuba / get
violin".
```

(The Paris Theatre in Lower Regent Street, London, was for many years the home of BBC radio panel games.)

ExampleAnteaters

The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing actions that apply to topics, as in "look up anteater colonies in the guide".

```
"Anteaters"


A book is a kind of thing. Understand "book" as a book.
A book has a table name called the contents.


Report consulting a book about:
    say "You flip through [the noun], but find no reference
to [the topic understood]." instead.


Instead of consulting a book about a topic listed in the
contents of the noun:
    say "[reply entry][paragraph break]".


The Guide to Desert Fauna is a book. The contents of
the Guide is the Table of Critters.
```

Table of Critters

reply
"You flip through the Guide for a while and eventually realise that spines are flora, not fauna."
"The giant anteater, which grows to six feet in size and can kill a jaguar, is a solitary animal, found in many habitats, including grasslands, deciduous forests and rainforests. It does not form colonies. That's ants. They're actually quite easy to tell apart."

Death Valley is a room. The Guide is in the Valley.

The gizmo is in Death Valley. The gizmo has an action called idea. The description of the gizmo is "The gizmo is hard to describe, but it projects an idea of [idea]."

Before when the player carries the gizmo and the idea of the gizmo is waiting:
    say "[The gizmo] eagerly soaks up the whole idea of [the current action].";
    now the idea of the gizmo is the current action.

After dropping the gizmo:
    say "The percussion of the fall seems to have shaken the gizmo's idea loose! There's nothing for it now but [idea of the gizmo].";
    try the idea of the gizmo;
    now the idea of the gizmo is waiting.

Test me with "get guide / look up spines in guide / x gizmo / get gizmo / i / x gizmo / drop gizmo / get gizmo / look up anteater colonies in guide / x gizmo / drop gizmo".

We can add the handling of multiple objects to an existing action simply by adding in a line of grammar using "[things]". In response, Inform will consider every object accepted by the token, and perform the action once for each of those objects. Thus:

```
"Shawn's Bad Day"

The Treasury is a room. The vault is a lockable locked
closed openable container in the Treasury. It is fixed in
place. "A massive vault fills up one wall." The
description is "The vault's system includes [a list of
things which are part of the vault]."

A little green light, a little blue light, a little red light, a
thin black pane of glass, a laser beam, a retinal
scanner, a thumbprint ID plate, a dial, and a large lever
are part of the vault.

The security guard is a man in the Treasury. The
description is "His name is Shawn, and he doesn't look
happy."

The description of the green light is "Off." The
description of the blue light is "Tranquilly on." The
description of the red light is "Angrily flashing."

Understand "examine [things]" as examining.
```

Test me with "examine all".

Example **The Best Till Last**

Reordering multiple objects for dramatic effect.

If a single command asks to do many things, some dull and some exciting, we may want to save the good ones for the end.

"The Best Till Last"

The Funky Ignition Lounge is a room. "This is where all evenings end." The stick of gelignite, the solid magnesium footstool, the vetiver candle, and the vodka bottle are here.

The burn description of the vetiver candle is "It burns right down, expensively but gothically."

The player carries an inexpensive firework. The description of the firework is "It is a cardboard tube with red and green stripes along the outside, and a fuse sticking out of the end." The burn description of the firework is "It ignites gloriously! You take a few hasty steps back in order to avoid burning yourself, and not a moment too soon. Red and green sparks fly out of the tube, and there's a whistling noise punctuated by several loud cracks."

The player carries a lighter. The description of the lighter is "You don't smoke, but you like to have access to flame now and then anyway."

```
Burning it with is an action applying to one thing and
one carried thing.


Understand "burn [things] with [something preferably
held]" as burning it with.


The block burning rule is not listed in any rulebook.


A thing has some text called the burn description.


Check burning something:
    if the player carries the lighter:
        try burning the noun with the lighter;
    else:
        try burning the noun with the noun.


Check burning something with something when the
second noun is not the lighter:
    say "Your trusty lighter is the best flame source
available to you." instead.


Check burning something with something:
    if the burn description of the noun is "":
        say "Best not." instead.


Carry out burning something with something:
    remove the noun from play.


Report burning something with something:
    say "[burn description of the noun][line break]".
```

```
A multiple action processing rule when the action name
part of the current action is the burning it with action
(this is the orderly burn rule):
    let L be the multiple object list;
    let dull list be a list of objects;
    let fun list be a list of objects;
    repeat with item running through L:
        if the burn description of the item is "":
            add item to dull list;
        else:
            add item to fun list;
    let F be the dull list;
    add fun list to F;
    alter the multiple object list to F.
```

```
Test me with "burn all with lighter".
```

ExampleWestern Art History 305

Allowing EXAMINE to see multiple objects with a single command.

In a gallery, there are many individual things to look at, but you can also get a general impression by just examining them as a collection.

First, we'll make a kind for the paintings exhibited in the gallery, and then we'll also make a special object to represent all of them as a mass:

```
"Western Art History 305"
```

```
A painting is a kind of thing. A painting is usually fixed in
place. Understand "painting" as a painting. Understand
"paintings" as the plural of painting.
```

```
The painting-collective is a thing. The printed name of
the painting-collective is "paintings". The description of
```

```
the painting-collective is "There's [a list of visible
paintings]."
```

We could if we wanted tweak the description to be different in style in different rooms of the gallery, but this will do for now. Next we need to make it possible to type something like EXAMINE PAINTINGS, which normally wouldn't work because the Standard Rules don't tell Inform to recognise multiple objects with the EXAMINE command (unlike, say, DROP or TAKE). This is easy:

```
Understand "examine [things]" as examining.
```

Now to make use of the special object. If the player types EXAMINE PAINTINGS, the multiple object list will become a list of the visible paintings. The following rule looks at this list: if it contains more than one painting, it replaces them with the painting-collective instead. Now there's only one examining action, so we get a reply like "There's an abstract painting, a pointilist painting and a French academic painting." instead of a list of descriptions of each in turn.

```
A multiple action processing rule when the current
action is examining (this is the examine kinds rule):
    let L be the multiple object list;
    let F be L;
    let the painting count be 0;
    repeat with item running through L:
        if the item is a painting:
            increment the painting count;
            remove the item from F;
    if the painting count is greater than one:
        add the painting-collective to F;
        alter the multiple object list to F.
```

And now some art to try this out on:

```
Gallery is a room. "Various paintings hang on the walls
of this gallery, awaiting critical attention. A side chamber
to the north contains smaller works."
```

```
The abstract painting, the pointilist painting, and the
French academic painting are paintings in the Gallery.
```

North of the Gallery is the Side Chamber. A handsome miniature is a painting in the Side Chamber. The description of the handsome miniature is "The miniature depicts a uniformed soldier of the late 18th century, with braid on his shoulders and a curl in his beard."

The player carries a small notebook. The description of the notebook is "It contains the notes you've taken so far towards a paper for Western Art History 305. So far you're still feeling a bit uninspired."

Test me with "x paintings / x all / n / x paintings / x all".

ExampleEscape from the Seraglio

Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

"Escape from the Seraglio"

Section 1 - Special Announcement Rules

The number of takes this turn is a number that varies. Every turn: now the number of takes this turn is 0.

The friskily announce items from multiple object lists rule is listed instead of the announce items from multiple object lists rule in the action-processing rules.

This is the friskily announce items from multiple object lists rule:
    if taking:
        if the current item from the multiple object list is not nothing:
            increment the number of takes this turn;
            say "[if number of takes this turn is 1]First [otherwise if the number of takes this turn is 2]And then [otherwise if the number of takes this turn is 3]And I suppose also [otherwise if the number of takes this turn is 7]And on we wearily go with [otherwise if the number of takes this turn is 9]Oh, and not forgetting [otherwise]And [end if][the current item from the multiple object list]: [run paragraph on]";
    otherwise:
        if the current item from the multiple object list is not nothing, say "[current item from the multiple object list]: [run paragraph on]".

Rule for deciding whether all includes the person asked: it does not.
Rule for deciding whether all includes a person when taking: it does not.

Section 2 - The Scenario

The Palm Chamber is a room. Sarissa is a woman in the Palm Chamber.

The Palm Chamber contains a bottle of ink, a quill pen, a tangerine, a bunch of grapes, a length of silken rope, some perfume, a cake of incense, a fitted leather bodice, a sapphire anklet, an illustrated novel, a whip, and a heavy iron key.

A persuasion rule for asking Sarissa to try taking the key:
    say "Sarissa nervously demurs, knowing that it is forbidden.";

persuasion fails.

A persuasion rule: persuasion succeeds.

Test me with "take all / drop all / sarissa, take all".

---

428 **ExampleFormicidae** *WI*

Manipulating the order in which items are handled after TAKE ALL.

Suppose we have an item that produces an interesting result the first time the player lifts it -- a rock with dangerous ants revealed underneath. The effect of the surprise is a little weakened, though, if the player sees that response as the result of a TAKE ALL, when it might be printed like this:

>[3] get all
tent peg: Taken.
water flask: Taken.
trading permit: Taken.
innocent-looking rock: You reach for the rock and turn it over to reveal a thriving colony of flesh-eating ants. Needless to say, you drop the rock and jump back with a decidedly effeminate scream. They can probably hear you all the way back in the base camp.
rusty nail: Taken.

[Your score has just gone down by two points.]

The calm response to "rusty nail" looks odd now, and the score change is disconnected from the event that caused it.

To manage this, we might institute a system so that interesting objects are handled last in their list, like so:

"Formicidae"

Use scoring.

Section 1 - Procedure

The magic rule is listed before the generate action rule in the turn sequence rules.

A thing has a number called dramatic potential.

This is the magic rule:
    let L be the multiple object list;
    if the number of entries in L is greater than 1:
        sort L in dramatic potential order;
        alter the multiple object list to L.

Section 2 - Scenario

The Foothills is a room. "The land has become hilly; though the soil is still mostly coarse yellow sand, clumps of grass are able to grow in the shadier places. Deep wagon ruts running from the southwest towards the mountains in the northeast show where generations of caravans have already passed."

The water flask, the tent peg, and the trading permit are things in Foothills.

```
The rock is a thing in Foothills. Before printing the name
of the rock when the rock is not handled: say "innocent-
looking ". The dramatic potential of the rock is 10.



The rusty nail is a thing in Foothills.



The ant colony is a fixed in place thing. "A busy group of
ants are crawling to and fro in the unaccustomed sun."
Rule for deciding whether all includes the ant colony
while taking: it does not.



Instead of taking the rock when the rock is handled:
    say "It might still have a stray ant or two on it."



After taking the rock:
    now the rock is handled;
    move ant colony to the location;
    move the rock to the location;
    say "You reach for the rock and turn it over to reveal
a thriving colony of flesh-eating ants. Needless to say,
you drop the rock and jump back with a decidedly
effeminate scream. They can probably hear you all the
way back in the base camp.";
    decrease score by 2.



Test me with "get peg / drop peg / get all / get rock".
```

Note that while one could also manipulate the object list to add or remove items at this stage, there's a simpler way to control what Inform considers "ALL" to mean in commands: see the activity "Deciding whether all includes" in the activities chapter.

Example **The Facts Were These**

Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT

or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

Occasionally it happens that we want to process an action on multiple items differently than we would if the player had just typed each of the individual actions separately. In this example, the reason is that we can only successfully GIVE items when their combined value passes a certain threshold amount; otherwise the recipient will reject them.

This works as an implementation of money, if we give value only to cash objects (though several other implementations of cash are available, most of which are simpler and more efficient). We could also imagine a mechanic like this being used for a bargaining or auction game as well, given a society that deals in objects rather than credits.

In order to consider all the items in the gift at once, we create an action that applies to multiple objects, but will in fact test the whole object collection during the first pass and print a definitive answer to whether the action succeeded. All subsequent times the game consults the rulebook will be stopped at the very beginning. No further processing will occur or output be printed.

```
"The Facts Were These"
```

```
Section 1 - Procedure
```

We start by creating the idea that everything in the game has a monetary value:

```
A price is a kind of value. $10 specifies a price. A thing
has a price.
```

```
Understand "give [things preferably held] to [someone]"
as multiply-giving it to. Understand "give [things] to
[someone]" as multiply-giving it to. Multiply-giving it to is
an action applying to two things.
```

A subtlety here: we say "things preferably held" to prefer items that the player is holding (so if the player has two dollars in hand and a third lies on the ground, he will use just the two he has).

The second grammar line allows Inform to match things that aren't held if it can't make up the list from things that are. If all three dollars are on the ground, the player can pick them up before spending them.

We do not, however, make multiply-giving apply to a "carried" item, because that will generate implicit takes of those items in a way that will mess up our action reporting. Instead, we're going to build the implicit takes into the system in a different way, one that permits us to collate the reports more attractively and print a short, one-sentence list of anything that the player had to pick up.

```
A thing can be given or ungiven. A thing is usually
ungiven.
```

This is for record-keeping purposes so that we can print an attractive list of what was given at the end of the turn.

```
First check multiply-giving it to:
    if already gave at the office is true:
        stop the action.


Already gave at the office is a truth state that varies.
```

"Already gave at the office" is the perhaps-excessively-named flag that keeps track of whether we've already done this action once.

```
Check multiply-giving something to the player:
        now already gave at the office is true;
        say "You can hardly bribe yourself.[paragraph
break]" instead;
```

The following rule is longish because it processes the entire list at once, generating implicit takes if necessary (but processing those implicit takes silently according to its own special rule, so that the output can be managed attractively). We are also, at the same time, calculating the total value of the player's offer.

```
Check multiply-giving it to:
    let L be the multiple object list;
```

```
    let bribe-price be $0;
    repeat with item running through L:
        if the player does not carry the item:
            abide by the ungivability rules for the item;
            carry out the implicitly taking activity with the
item;
            if the player does not carry the item:
                now already gave at the office is true;
                say "You can't include [the item] in your bribe,
since you're not holding [them]![paragraph break]"
instead;
        increase bribe-price by the price of item;
    if the number of entries in the recently-collected list is
greater than 0:
        repeat with item running through the recently-
collected list:
            now item is marked for listing;
        say "You pick up [the list of marked for listing
things] and make your offer. [run paragraph on]";
        now everything is unmarked for listing;
    if the bribe-price is less than the price of the second
noun:
        now already gave at the office is true;
        say "[The second noun] angrily rejects your piffling
bribe.[paragraph break]" instead.
```

The bit about making some items "marked for listing", above, rather than printing the list directly, is that using the "[the list of....]" syntax guarantees that Inform will respect grouping rules in writing its description. For instance, if the player has automatically taken all three dollars, the output will say "the three dollars" instead of "the dollar, the dollar, and the dollar."

```
Carry out multiply-giving it to:
    let L be the multiple object list;
    repeat with item running through L:
        now the second noun carries the item;
        now the item is given;
    now already gave at the office is true;



Report multiply-giving it to:
    say "[The second noun] rather shamefacedly tucks
[the list of given things] away into a pocket.[paragraph
break]".
```

Now we create our own variation of implicitly taking in order to customize

the output for the multiply-giving action. The "ungivability rules" should disallow any object that the player absolutely cannot take, because we want "carry out the implicitly taking activity" to succeed every time -- and therefore not print out any less-attractive results from implicit takes that don't succeed. Otherwise, the player's GIVE TREE AND DOG TO ATTENDANT might produce the reply "That's fixed in place" -- without specifying which object is fixed in place.

Because of the way this works, we will want to be careful: if we have any "instead of taking..." rules for special objects in the game, we should be sure to mirror those with an ungivability rule to print something more suitable in the case that the player tries taking that object as part of the multiple giving action.

```
The ungivability rules are an object-based rulebook.


An ungivability rule for a person:
    now already gave at the office is true;
    say "Slavery is illegal.[paragraph break]" instead.


An ungivability rule for something (called the item) which
is enclosed by someone who is not the player:
    now already gave at the office is true;
    say "[The item] [aren't] yours to give.[paragraph
break]" instead.


An ungivability rule for something which encloses the
player:
    now already gave at the office is true;
    say "You don't want to end up as part of the gift.
[paragraph break]" instead;


An ungivability rule for something (called the item) which
is part of something:
    now already gave at the office is true;
    say "[The item] [are] attached to [a random thing
which incorporates the item][paragraph break]" instead.
```

```
An ungivability rule for something (called the item) which
is scenery:
    now already gave at the office is true;
    say "[The item] [are] unremovable.[paragraph break]"
instead.


An ungivability rule for something (called the item) which
is fixed in place:
    now already gave at the office is true;
    say "[The item] [are] fixed in place.[paragraph break]"
instead.


An ungivability rule for a direction (called the item):
    now already gave at the office is true;
    say "[The item] [are] not susceptible to giving.
[paragraph break]" instead.


Rule for implicitly taking something (called target) while
multiply-giving:
    silently try taking the target;
    if the player carries the target:
        add the target to the recently-collected list.


The recently-collected list is a list of objects that varies.
```

And since we don't want to list the individual objects separately:

```
The selectively announce items from multiple object lists
rule is listed instead of the announce items from multiple
object lists rule in the action-processing rules.


This is the selectively announce items from multiple
object lists rule:
    if multiply-giving:
```

```
        do nothing;
    otherwise:
        if the current item from the multiple object list is not
nothing:
            say "[current item from the multiple object list]:
[run paragraph on]".
```

And now, since this ought to work symmetrically if the player provides just one high-value item:

```
Check giving something to someone:
    if the price of the noun is less than the price of the
second noun:
        say "[The second noun] angrily rejects your piffling
bribe." instead.
```

As we've seen elsewhere, the giving action by default returns a refusal, but is also written to start working if we remove the blockage. So we do that here, and revise the report rule to match the report rule we have for multiple giving.

```
The block giving rule is not listed in any rulebook.

The new report giving rule is listed instead of the
standard report giving rule in the report giving it to rules.

This is the new report giving rule:
    say "[The second noun] rather shamefacedly tucks
[the noun] away into a pocket."
```

After each instance of the multiply-giving action, we need to clear the variables we used to track its state. We could do this in "Before reading a command", but that's unsafe because the player might type GIVE PIE AND CAP TO ATTENDANT. GIVE DOLLARS TO ATTENDANT. all on a single line, and we would like to be able to clear the variables between one action and the next. The correct place to attach this behavior is immediately before the generate action rule, thus:

```
The before-generation rule is listed before the generate
```

action rule in the turn sequence rules.

This is the before-generation rule:
    now every thing is ungiven;
    now already gave at the office is false;
    truncate the recently-collected list to 0 entries.

Section 2 - Scenario

The Morgue Office is a room. "This is not the Morgue itself; this is only its outer office. The familiar room full of silver drawers and cold air lies beyond."

The Morgue Attendant is a man in the Morgue Office. "The Attendant has seen you come through a number of times, and is becoming suspicious of your abiding interest in dead people." The description is "The Morgue Attendant is fifty-four years, six months, five days, and three minutes old." The price of the Morgue Attendant is $3.

A dollar is a kind of thing. The player carries three dollars. The price of a dollar is always $1.

The player carries a miniature rhubarb pie. The price of the miniature rhubarb pie is $5.

The player carries a knitted cap. The price of the knitted cap is $2.

Test me with "test dollars / purloin three dollars / test multi-line / purloin three dollars / purloin pie / purloin cap

/ test specificity / purloin three dollars / test largesse / test mixed-gift".

Test multi-line with "give dollar and pie to attendant. give dollars and cap to attendant".

Test dollars with "drop all / give dollar to Morgue Attendant / give dollars to Morgue Attendant / get dollars / give dollars to morgue attendant / purloin three dollars / drop dollars / give dollars to Morgue Attendant".

Test specificity with "give three dollars to Morgue Attendant".

Test largesse with "give pie to Morgue Attendant".

Test mixed-gift with "give dollar and cap to Morgue Attendant / get cap / give dollar and cap to morgue attendant / give me and dollar to attendant".

PURLOIN, used in the tests here, is a special debugging command that allows the player to acquire objects that wouldn't otherwise be possible to take. It is only active in non-release versions of the story. For more about debugging commands, see the chapter on Testing and Debugging.

Example **Alpaca Farm**

A generic USE action which behaves sensibly with a range of different objects.

This example takes the ordering of grammar lines to its logical extreme, sorting the player's input into different categories depending on the kind and condition of the objects mentioned.

"Alpaca Farm"

Understand "use [an edible thing]" as eating.

Understand "use [a wearable thing]" as wearing.

Understand "use [a closed openable container]" as opening. Understand "use [an open openable container]" as closing.

Understand "use [something preferably held] on [a locked lockable thing]" as unlocking it with (with nouns reversed). Understand "use [something preferably held] on [an unlocked lockable thing]" as locking it with (with nouns reversed).

Understand "use [a switched off device]" as switching on.

Understand "use [something]" as using. Using is an action applying to one thing. Carry out using: say "You will have to be more specific about your intentions."

Understand "use [a door]" as opening. Understand "use [an open door]" as entering.

The Llama Pen is a room. North of the Pen is the gate. The gate is a door. North of the gate is the Rocky Path. The brown llama is an animal in the Llama Pen.

Appearance is a kind of value. The appearances are muddy, scruffy, fluffy, and dapper. The brown llama has an appearance. The brown llama is muddy. Before printing the name of the brown llama, say "[appearance] ". Before printing the name of the brown llama while grooming: say "now-[if appearance of the brown llama is less than dapper]merely-[end if]".

A grooming tool is a kind of thing. Understand "use [a grooming tool] on [something]" as grooming it with (with nouns reversed). Grooming it with is an action applying to two things. Understand "groom [something] with [something]" as grooming it with.

Carry out grooming it with:
    if the appearance of the noun is less than dapper, now the appearance of the noun is the appearance after the appearance of the noun.

Report grooming it with:
    say "You attend diligently to the appearance and hygiene of [the noun]."

Instead of using a grooming tool in the presence of the brown llama:
    try grooming the brown llama with the noun.

The player carries some nail nippers, a slicker brush, and an apple. The apple is edible. The brush and the nippers are grooming tools. The player wears a sombrero.

The description of the nail nippers is "Ten inches long, to give you the necessary leverage to cut tough llama toenails. It still helps to soften them up by making the llama stand in a bucket of water first, though."

The description of the slicker brush is "Fine, angled soft bristles set into a broad back, perfect for removing mud from the coat of a long-woolled llama."

The industrial-strength blower is a fixed in place device in the Llama Pen. "Attached to the nearest wall, on its own movable boom, is an industrial-strength blower for doing llama hair."

Understand "use [switched off blower]" as switching on. Understand "use [switched on blower] on [brown llama]" as grooming it with (with nouns reversed). Instead of using the blower in the presence of the brown llama, try grooming the brown llama with the blower.

Test me with "use gate / use blower / use nippers / use brush / use apple / remove sombrero / use sombrero".

Whether we actually want a USE action is a subject of some theoretical debate in the IF community. On the one hand, it helps avoid guess-the-verb problems where the player cannot figure out what term to use in order to express a fairly simple idea. On the other, it encourages the player to think that all items have one and exactly one use, rather than getting him to consider the range of possibilities that arise from having a complex vocabulary.

ExampleApples

Prompting the player on how to disambiguate otherwise similar objects.

Inform by default detects whether two objects can be disambiguated by any vocabulary available to the player. If so, it asks a question; if not, it picks one of the identical objects at random.

Generally this produces good behavior. Occasionally, though, two objects have some distinguishing characteristic that doesn't appear in the

object name. For instance, suppose we've created a class of apples that can be told apart depending on whether they've been bitten or not:

```
An apple is a kind of thing. Consumption is a kind of
value. The consumptions are pristine and bitten. An
apple has a consumption. The description of an apple is
"It is [consumption]."


Understand the consumption property as describing an
apple.
```

The player can meaningfully type

```
>EAT BITTEN APPLE
```

or

```
>EAT PRISTINE APPLE
```

but if he types

```
>EAT APPLE
```

Inform will, annoyingly, ask

```
Which do you mean, an apple or the apple?
```

This gives the player no indication of why Inform is making a distinction. So here we add a special "printing the name" rule to get around that situation:

```
"Apples"
```

```
Orchard is a room.


An apple is a kind of thing. Consumption is a kind of
value. The consumptions are pristine and bitten. An
apple has a consumption. The description of an apple is
"It is [consumption]."


Understand the consumption property as describing an
apple.


Before printing the name of an apple while asking which
do you mean: say "[consumption] ". Before printing the
plural name of an apple while asking which do you
mean: say "[consumption] ".


The player carries three apples.


Instead of eating a pristine apple (called the fruit):
    say "You take a satisfying bite.";
    now the fruit is bitten.


Instead of eating a bitten apple (called the fruit):
    say "You consume the apple entirely.";
    now the fruit is nowhere.
```

Inform will also separate the bitten from the pristine apples in inventory listings and room descriptions, even though it's not clear why; we can improve on that behavior thus:

```
Before listing contents: group apples together.
```

```
Rule for grouping together an apple (called target):
    let source be the holder of the target;
    say "[number of apples held by the source in words]
apple[s], some bitten".


Before printing the plural name of an apple (called
target):
    let source be the holder of the target;
    if every apple held by the source is bitten, say "bitten
";
    if every apple held by the source is pristine, say
"pristine ".


Test me with "i / eat apple / i / eat apple / pristine / i / eat
apple / pristine / i".
```

The parser error "That noun did not make sense in this context" arises instead of "You can't see any such thing" when the player uses a command that could apply to any item in the game -- that is, a command such as

```
Understand "go to [any room]" as going directly to.
Understand "talk about [any subject]" as discussing.
```

...and so on. The idea here is that "You can't see any such thing" isn't a sensible rejoinder when the player doesn't really need to be able to see the object.

Nonetheless, "That noun did not make sense..." is itself a fairly dry and uninformative response, and we may want to override it to something more appropriate for the specific kind of context in which it might appear. For instance:

```
"WXPQ"

WXPQ Studio is a room. "After about 2 AM, no one is
listening anyway, so you can more or less make up
whatever you like to fill the airwaves."

John F Kennedy, Elvis, Ralph Nader, Tony Blair, and
single-origin chocolate are things.

Understand "talk about [any thing]" or "discuss [any
thing]" as discussing. Discussing is an action applying to
one visible thing.

Carry out discussing:
    say "You babble for a while about your [one
of]interest in[or]hatred of[or]passionate devotion
to[or]conspiracy theory concerning[or]mother's secret
love affair with[as decreasingly likely outcomes] [the
noun]."

Rule for printing a parser error when the latest parser
error is the noun did not make sense in that context
error:
    say "For once, you're at a loss for anything to say."

Test me with "discuss Elvis / discuss Kennedy / discuss
chocolate / discuss narratology vs ludology debate".
```

Note that this solution works as simply as it does because we only have one command in the game that can apply to an "[any]" token. If we had several, we'd need to distinguish between the parser error attached to "discuss" and the parser error attached to "go to" (for instance). In that case, we might instead write something like

```
Rule for printing a parser error when the latest parser
error is the noun did not make sense in that context
error:
if the player's command includes "go":
    say "There's no such place you know how to get to.";
otherwise:
    say "For once, you're at a loss for anything to say."
```

ExampleWalls and Noses *WI*

Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

Suppose we want our game to respond to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

For the case of EXAMINE WALL, we need a way to determine whether every item being disambiguated is a direction. We'll start by making a "matched" adjective which will identify items being disambiguated:

```
"Walls and Noses"

Eight-Walled Chamber is a room. "A perfectly octagonal
room whose walls are tinted in various hues."

Understand "wall" as a direction.

Definition: a direction is matched if it fits the parse list.
Definition: a room is matched if it fits the parse list.
Definition: a thing is matched if it fits the parse list.

Rule for asking which do you mean when everything
matched is direction:
    say "In which direction?"
```

Checking the parse list requires a bit of behind-the-scenes work with Inform 6. Fortunately, you don't have to understand this entirely in order to use the rest of the example:

```
To decide whether (N - an object) fits the parse list:
    (- (FindInParseList({N})) -)



Include (-
[ FindInParseList obj i k marker;
    marker = 0;
    for (i=1 : i<=number_of_classes : i++) {
    while (((match_classes-->marker) ~= i) &&
((match_classes-->marker) ~= -i)) marker++;
    k = match_list-->marker;
    if (k==obj) rtrue;
    }
    rfalse;
];
-)
```

Now that we've defined our "matched" adjective, we can use it for other purposes as well -- even generating our own lists. Our second challenge was to respond to EXAMINE NOSE with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

Here we need to change the way the question is worded (not "which do you mean" but "whose nose do you mean"). We also have to the names of the noses as they're printed in this particular context, so that they don't repeat the word "nose" over and over. And -- as a point of good English style -- we also want "your own" nose always to be last on the list.

For this purpose we may want to use the built-in "Complex Listing" extension, which allows us to print specially ordered lists. So:

```
Include Complex Listing by Emily Short.



Wilma, Betty, and Frederica are women in the Eight-
Walled Chamber. Understand "lady" or "woman" as a
woman. A nose is a kind of thing. A nose is part of every
person.
```

```
Rule for asking which do you mean when everything
matched is a nose:
    prepare a list of matched things;
    if your nose is an output listed in the Table of Scored
Listing:
        choose row with an output of your nose in the
Table of Scored Listing;
        now the assigned score entry is -1;
    say "Whose nose do you mean, [the prepared list
delimited in disjunctive style]?"


Rule for printing the name of a nose (called target) while
asking which do you mean :
    if everything matched is a nose:
        if the target is part of a person (called owner):
            if the owner is the player, say "your own";
            otherwise say "[the owner][apostrophe]s";
    otherwise:
        make no decision.


Understand "own" or "mine" as your nose.


Test me with "x wall / north / x nose / mine".
```

Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

Novice players of interactive fiction, unfamiliar with its conventions, will often try to add extra phrases to a command that the game cannot properly parse: HIT DOOR WITH FIST, for instance, instead of HIT DOOR.

While we can deal with some of these instances by expanding our range of actions, at some point it becomes impossible to account for all the possible prepositional phrases that the player might want to tack on. So

what do we do if we want to handle those appended bits of text sensibly?

We could go through and remove any piece of text containing "with ..." from the end of a player's command; the problem with that is that it overzealously lops off the ends of valid commands like UNLOCK DOOR WITH KEY, as well. So clearly we don't want to do this as part of the "After reading a command..." stage.

A better time to cut off the offending text is right before issuing a parser error. At that point, Inform has already determined that it definitely cannot parse the instruction as given, so we know that there's something wrong with it.

The next problem, though, is that after we've edited the player's text we want to feed the corrected version back to Inform and try once more to interpret it.

This is where we have a valid reason to write a new "rule for reading a command". We will tell Inform that when we have just corrected the player's input to something new, it should not ask for a new command (by printing a prompt and waiting for another line of input); it should instead paste our stored corrected command back into "the player's command" and proceed as though that new text had just been typed.

Thanks to John Clemens for the specifics of the implementation.

```
"Cave-troll" by JDC


Section 1 - The Mechanism


The last command is a text that varies.


The parser error flag is a truth state that varies. The
parser error flag is false.


Rule for printing a parser error when the latest parser
error is the only understood as far as error and the
player's command matches the text "with":
    now the last command is the player's command;
```

```
    now the parser error flag is true;
    let n be "[the player's command]";
    replace the regular expression ".* with (.*)" in n with
"with \1";
    say "(ignoring the unnecessary words '[n]')[line
break]";
    replace the regular expression "with .*" in the last
command with "".



Rule for reading a command when the parser error flag
is true:
    now the parser error flag is false;
    change the text of the player's command to the last
command.



Section 2 - The Scenario



The Cave is a room.



The troll is a man in the cave.



The player carries a sword.



The chest is a locked lockable container in the cave.



Test me with "attack troll with sword / unlock chest with
sword / attack troll as a test".
```

A caveat about using this method in a larger game: "parser error flag" will not automatically control the behavior of any rules we might have written for Before reading a command... or After reading a command..., so they may now fire at inappropriate times. It is a good idea to check for parser error flag in those rules as well.

ExampleDown in Oodville

Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

Now and then in IF there is a situation where we need to ask the player for a numbered choice rather than an ordinary action command. What's more, that numbered choice might change during the game, so we don't want to just hard-wire the meanings of "1", "2", and "3" whenever the player types them.

A better trick is to keep a list or table (we'll use a table here because it involves slightly less overhead) recording what the player's numerical choices currently mean. Then every time the player selects a number, the table is consulted, and if the number corresponds to something, the player's choice is acted on.

In our example, we'll have a transporter pad that can take the player to any room in the game that he's already visited. (Just for the sake of example, we'll start him off with a few pre-visited rooms.)

```
"Down in Oodville"


Section 1 - Method


Understand "[number]" as selecting.


Selecting is an action applying to one number.


Check selecting: [assuming we don't want to be able to
transport from just anywhere]
    if the player is not on the transporter pad:
        say "You can transport only from the transporter
pad. From other places than the transporter room, you
can HOME to your base ship, but not leap sideways to
```

```
other locations.";
        empty the transport options instead.



Check selecting:
    if the number understood is greater than the number
of filled rows in the Table of Transport Options or the
number understood is less than one:
        say "[The number understood] is not a valid option.
";
        list the transport options instead.



Carry out selecting:
    let N be the number understood; [not actually a
necessary step, but it makes the next line easier to
understand]
    choose row N in the Table of Transport Options;
    if the transport entry is a room:
        move the player to the transport entry;
    otherwise:
        say "*** BUG: Improperly filled table of transport
options ***" [It should not be possible for this to occur,
but we add an error message for it so that, if it ever
does, we will know what is causing the programming
error in our code]



To list the transport options:
    let N be 1;
    say "From here you could choose to go to: [line
break]";
    repeat through the Table of Transport Options:
        say " [N]: [transport entry][line break]";
        increment N.



To empty the transport options:
    repeat through the Table of Transport Options:
        blank out the whole row; [first we empty the table]



To load the transport options:
    repeat with interesting room running through visited
```

rooms which are not the Transporter Room:
	choose a blank row in the Table of Transport Options;
	now the transport entry is the interesting room.

Table of Transport Options
transport
an object
with 3 blank rows. [In the current scenario, the number of blank rows need never be greater than the number of rooms in the game, minus the transport room itself.]

Understand "home" as homing. Homing is an action applying to nothing.

Check homing:
	if the player is in the Transporter Room:
		say "You're already here!" instead.

Carry out homing:
	move the player to the transporter room.

Section 2 - Scenario

The Transporter Room is a room.

Oodville is a visited room.

Midnight is a visited room. The Diamond City is west of Midnight.

```
The transporter pad is an enterable supporter in the
Transporter Room. "The transporter pad in the middle of
the floor is currently dull blue: powered but unoccupied."


After entering the transporter pad:
    say "The transporter beeps and glows amber as you
step onto its surface. A moment later a hologram
displays your options. [run paragraph on]";
    empty the transport options;
    load the transport options;
    list the transport options.


Test me with "get on pad / 0 / -1 / 8 / 2 / look / w / home
/ get on pad / get off pad / 3".
```

If we wanted to replace the regular command structure entirely with numbered menus, or use menus to hold conversation options, we could: several Inform extensions provide these functions.

---

373 **ExampleCloves** *WI*

Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

It has sometimes been suggested that IF should allow for the player to use adverbs, so that doing something "carefully" will have a different effect from doing it "quickly". There are several inherent challenges here: it's a good idea to make very sure the player knows all his adverb options, and the list of possibilities should probably not be too long.

Another trick is that adverbs complicate understanding commands, because they can occur anywhere: one might type >GO WEST CAREFULLY or >CAREFULLY GO WEST, and ideally the game should understand both. After reading a command is the best point to do this sort of thing, because we can find adverbs, interpret them, and remove them from the command stream. So:

"Cloves"

Manner is a kind of value. The manners are insouciantly, sheepishly, and defiantly.

Now we have, automatically, a value called manner understood to be used whenever parsing manners, and we can use this even during the "after reading a command" stage, so:

```
After reading a command:
    if the player's command includes "[manner]":
        cut the matched text;
    otherwise:
        say "But how, my dear boy, how? You simply can't
do something without a pose. Thus far you have
mastered doing things [list of manners].";
        reject the player's command.
```

```
When play begins:
    now the left hand status line is "Behaving [manner
understood]";
    now the right hand status line is "[location]";
    now the manner understood is insouciantly.
```

```
The Poseur Club is a room. "Lady Mary is laid out on a
sofa, her wrists bandaged importantly[if the manner
understood is insouciantly] -- and she looks all the more
depressed by your indifference to her state[end if];
Salvatore is at the gaming table, clutching his hair with
both hands[if the manner understood is defiantly] --
though he looks up long enough to snarl in response to
that expression of yours[end if]; Frackenbush is
muttering lines from another of his works in progress, as
though poetry has nearly made him mad[if the manner
understood is sheepishly]. But he spares you a
reassuring smile. He's not a bad fellow,
Frackenbush[end if].

The usual people, in short."
```

Instead of doing something other than waiting or looking:
    say "Dear. No. That would smack of effort."

Instead of waiting when the manner understood is sheepishly:
    say "You scuff your foot against the ground for a moment, and allow a seemly blush to creep over your cheek. It's quite effective, you are sure, though you can't look up and see how it is going."

Instead of waiting when the manner understood is insouciantly:
    say "Thrusting your hands into your pockets, you whistle a jaunty tune.

'Do shut up,' says a Melancholy Poseur from over by the window."

Instead of waiting when the manner understood is defiantly:
    say "You raise your chin and give a pointed glance around the room as though to say that you are waiting for someone; you are unembarrassed about waiting for her; you have by no means been stood up; and the first person to comment will receive a poke in the eye."

Before looking when the manner understood is sheepishly:
    say "You gaze up from under your brows..."

Before looking when the manner understood is defiantly:
    say "You cast a withering gaze over the room."

```
Before looking when the manner understood is
insouciantly:
    if turn count > 1,
        say "You turn an eye to your surroundings, looking
faintly-- just faintly-- amused."



Test me with "wait / wait insouciantly / sheepishly look /
defiantly look / look insouciantly".
```

The qualification about turn count is to prevent this before message from occurring when the player first looks around the room (automatically) at the start of play.

Note that to test this example, one must type INSOUCIANTLY TEST ME, and not simply TEST ME: a poseur's work is never done.

ExampleFragment of a Greek Tragedy

Responding to the player's input based on keywords only, and overriding the original parser entirely.

Apologies to the shade of A. E. Housman.

```
"Fragment of a Greek Tragedy"



Understand "restart/restore/save/quit" as "[meta-
command]".



After reading a command:
    if the player's command matches "[meta-command]",
make no decision;
    say line break;
    repeat through Table of Current Topics:
        if the player's command includes topic entry:
            say "CHORUS: [reply entry][paragraph break]";
```

```
        follow the advance time rule;
        rule succeeds;
    say "[italic type] Pause.[roman type][line break]";
    follow the advance time rule;
    rule succeeds.
```

```
Table of Current Topics

topic                                   reply
"Sailing trip/travel/cruise/yacht"      "Sailing ships have no need for oars."
"Rowing/oars/galley"                    "Rowing is for wimps."
"Mythology/gods/heroes/Zeus"            "Shut up, you whoreson Zeus!"
```

This would be a bit bare if we didn't provide the player with some sort of context at the outset, so let's put some remarks before the first command prompt:

```
Before reading a command while the turn count is 1:
    say "CHORUS: O suitably-attired-in-leather-boots
    Head of a traveller, wherefore seeking whom
    Whence by what way how purposed art thou come
    To this well-nightingaled vicinity?
    My object in inquiring is to know.
    But if you happen to be deaf and dumb
    And do not understand a word I say,
    Then wave your hand, to signify as much."
```

This "turn count" condition is why it was useful to follow the advance time rule in "after reading a command": the game (or drama, if you like) will continue to count moves elapsed even though the rest of Inform's command parsing and world model is being ignored. In a longer and more ambitious implementation of this idea, we might want to allow scenes to govern the behavior and responses of the Chorus.

And then to give the whole exchange a play's format:

```
The Stage is a room.
```

```
The room description heading rule is not listed in the
carry out looking rules.
```

```
When play begins:
    now the command prompt is "YOU: ";
    now left hand status line is "Fragment of a Greek
Tragedy";
    now right hand status line is "A. E. Housman".
```

(Because this example manipulates commands outside of the normal parser, the mechanism for TEST will not work here. Try typing commands such as: TELL CHORUS ABOUT JOURNEY / TELL CHORUS ABOUT FEET / TELL CHORUS ABOUT SHROPSHIRE / TELL CHORUS ABOUT ZEUS)

Chapter 7: Other Characters

Contents of *The Inform Recipe Book*
Chapter 6: Commands
Chapter 8: Vehicles, Animals and Furniture
Indexes of the examples

§7.1. Getting Acquainted

Talking about characters presents some special challenges. For one thing, some characters are referred to by a proper name, but others are not: so the story might want to talk about "Jack" but also about "the drunk pedestrian". In the absence of other information, Inform attempts to divine our intentions based on the words with which we defined a new character: but we can always override its guess with an explicit statement, such as

```
The Great Malefactor is proper-named.
```

Belfry demonstrates further how titles are set at the start of play.

The relation between the player and the other characters is not always static, however. Sometimes we want the player to learn a character's name part-way through play, and start referring to "the drunk pedestrian" as "Fernando". Similarly, the status of another character may change due to some twist of the plot. **Gopher-wood** shows how to change the name of a character mid-story, and **Peers** handles changing the character's rank.

Alternatively, of course, the player character may already know some of the other characters when the story begins, even if the player does not. In that case, we may want to add a tag-line or so of identification to a character's name when he first appears in the story. **A Humble Wayside Flower** shows one way of doing this.

Another occasional challenge is dealing with such commands as EXAMINE DR. THISBY. The problem here is that Inform by default will understand the full stop after "Dr" to be the end of one command and the beginning of another, and will try to interpret "Thisby" as a verb. If we do have a story populated by such formally-

addressed characters, we may turn to Punctuation Removal, which provides a phrase to remove the full stops in standard titles before attempting to interpret the command.

Other characters have physical characteristics as well as names, of course, and **Meet Market** demonstrates one way of implementing people with notable features.

Finally, in some IF, the roles of characters may change from playing to playing. If we are writing a replayable murder mystery, we might want to select a new culprit each time the story starts; for this, see **Clueless**.

See **The Human Body** *for more on body parts and physical description*

See **Memory and Knowledge** *for a way to refer to characters whom the player knows about but who aren't currently in the room*

Start of Chapter 7: Other Characters

Back to Chapter 6: Commands: §6.18. Alternatives To Standard Parsing

Onward to §7.2. Liveliness

Example 31: ★ **Belfry** You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

Example 135: ★ **Clueless** A murderer for the mystery is selected randomly at the beginning of the game.

Example 238: ★ **Meet Market** A case in which relations give characters multiple values of the same kind.

Example 32: ★★ **Gopher-wood** Changing the name of a character in the middle of play, removing the article.

Example 315: ★★ **Peers** The peers of the English realm come

in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince - and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

Example 237: ★★★ **A Humble Wayside Flower** Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

---

§7.2. Liveliness

A character who sits still in a chair and does nothing is much less convincingly alive than one who seems to be pursuing some sort of personal agenda. There are all sorts of ways to achieve this, but the least challenging is by introducing some random change to descriptions, and by giving a character some very simple routine behavior to carry out.

For instance, we'll often want the characters in a room to be described doing different things every time we look at them. **Camp Bethel** shows how this may be done.

Every turn rules lend some sprightliness to otherwise-silent characters, as well:

```
Every turn when the player can see Mrs MacGillicuddy:
    say "Mrs. MacGillicuddy vacuums around [a random fixed in place
thing which is in the location]."
```

We might expand on this by providing a whole table of things for Mrs MacG. to cycle through, or a text variation effect that gives her different activities every turn, as in

```
Every turn when the player can see Mrs MacGillicuddy:
    say "Mrs. MacGillicuddy [one of]vacuums around the
furniture[or]tries to remove gum from the underside of the
desks[or]causes a racket by testing the smoke alarm[or]makes a
```

> pointed comment or two about them as sit by idly while someone works her fingers to the bone[as decreasingly likely outcomes]."

This is no great innovation in characterization by itself, but it does at least remind the player that the character is alive and moving about, even when he isn't paying attention to her.

Annoyotron Jr demonstrates a character who actively tries to get our attention, and whose routine of behavior changes just slightly if we show signs of having reacted to him.

Lean and Hungry implements a classic thief, a character who doesn't interact with the player much except to pick up valuable objects he finds left around the map. Later we will see rather more advanced ways to make characters act on their own goals and plans, but this kind of simple behavior is easily rigged as part of an every turn rule.

Finally, **Text Foosball** extends the every-turn-rule idea to create an opponent who joins us in a randomized game of table soccer.

With animal characters, this kind of repetitive, semi-random behavior is often adequate: we don't expect animals to talk, or pursue steady goals, or to take an interest in what we do in their presence (unless it involves a food they like to eat).

For people, we are likely to need an assortment of additional techniques.

See **Animals** *for a domestic cat and dog*

Start of Chapter 7: Other Characters

Back to §7.1. Getting Acquainted

Onward to §7.3. Reactive Characters

Example 153: ★ **Annoyotron Jr** A child who after a certain period in the car starts asking annoying questions.

Example 67: ★★ **Camp Bethel** Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

Example 76: ★★★ **Lean and**

Hungry A thief who will identify and take any valuable thing lying around that he is able to touch.

Example 140: ★★★★ **Text Foosball** A game of foosball which relies heavily on every-turn rules.

---

§7.3. Reactive Characters

As we observe characters, so they observe us. Those who seem to have no awareness of what the player is doing often come across more like waxworks than like people. **Zodiac** demonstrates a scenario where the watchful presence of a dangerous criminal keeps the player from doing what he otherwise might, while **Police State** expands on this idea with a policeman who reacts to entire types of behavior in his presence, regardless of whether the culprit is the player or a third party. **Noisemaking** has a crow who will fly away in response to any loud noises the player makes.

And, of course, we definitely want to have characters react to being looked at or otherwise interfered with. **Search and Seizure** implements a smuggler who reacts when we try to confiscate his possessions. **Pine 1** gives us a sleeping princess who can be woken by a variety of methods.

We wrap up this section with two complete puzzle scenarios that demonstrate what can be achieved by giving characters reactions to the player's behavior. **A Day For Fresh Sushi** has a fish who watches the player's actions and comments on them, while the live furnishings in **Revenge of the Fussy Table** instead comment every turn on the current state of the world, until the player has successfully sorted out all their complaints.

Start of Chapter 7: Other Characters

Back to §7.2. Liveliness

Onward to §7.4. Barter and Exchange

Example 96: ★ **Zodiac** Several variations on "doing something other than...", demonstrating different degrees of restriction.

Example 154: ★ **Pine 1** Pine:

Using a scene to watch for the solution of a puzzle, however arrived-at by the player.

Example 187: ★ **Police State** Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

Example 38: ★★★ **Search and Seizure** A smuggler who has items, some of which are hidden.

Example 197: ★★★ **Noisemaking** Creating a stage after the report stage of an action, during which other characters may observe and react.

Example 82: ★★★★ **Revenge of the Fussy Table** A small game about resentful furniture and inconvenient objects.

Example 112: ★★★★★ **A Day For Fresh Sushi** A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is performed.

---

§7.4. Barter and Exchange

By default, Inform characters are a bit grudging about giving and sharing objects: they react with disinterest when they're shown things and refuse everything they're offered.

If we'd like to change this, we can simply remove the default "block giving" rule, as in

```
The block giving rule is not listed in the check giving it to rules.
```

If we do this, giving items to characters will have the result of moving our possessions to the other person's inventory. Of course, without more customization, the player may not ever be able to persuade the other character to return his possessions. **Bribery** demonstrates a scenario in which a character will accept gifts if they are interesting to him, and respond with a changed attitude to the player.

Barter Barter expands further on this by allowing other characters to trade things with one another.

See **Modifying Existing Commands** *for ways to allow the player to give or show things that he isn't currently holding*

See **Actions on Multiple Objects** *for an implementation of giving that allows the player to offer multiple objects at once, where their combined value determines whether they are accepted*

See **Money** *for ways to keep track of cash flow, physical money objects, and price negotiations*

Start of Chapter 7: Other Characters

Back to §7.3. Reactive Characters

Onward to §7.5. Combat and Death

Example 392: **Bribery** A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

Example 207: **Barter Barter** Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

---

§7.5. Combat and Death

Not all characters are friendly, and there are times when we may want to include a fight sequence. There are a number of ways to approach this, depending on whether we want to offer the player a random outcome, a predetermined one, or a combat sequence that depends partly on strategy or on having the proper equipment.

Lanista 1 demonstrates randomized combat in the style of a role-playing game. The player has a partially random chance of doing any given amount of damage; both the player and his opponent have hit points, and whichever one runs out first dies. **Lanista 2** continues this idea, but includes weapons that affect the amount of of damage done. **Red Cross** by itself implements a command that we might use to find out how strong characters are at the moment.

A word of warning about designing such sequences: a player who gets a roll he doesn't like always has the option of UNDOing a turn and re-rolling. This means that he can always win a random battle sooner or later; bad luck only means that it takes him longer (so he gets more bored and irritated as he plays through). It is possible to turn off UNDO implementation with

```
Use UNDO prevention.
```

...but there is a good chance that this will irritate players in itself. Role-playing-style combat scenarios need careful design, lest they actively make a story less fun.

In a slightly more realistic setting, combat leaves physical remains behind, unless we're wielding some kind of futuristic weapon that evaporates our opponents entirely: **Puff of Orange Smoke** demonstrates characters who leave corpses behind when they die, while **Technological Terror** more tamely explodes robots into numerous component parts.

Finally, we can imagine some scenarios in which, instead of allowing characters to strike at each other for random damage, we want to introduce an element of strategy. **Don Pedro's Revenge** shows the rudiments of a system in which the characters can make different kinds of attack depending on where they are in a room filled with perches, barrels, and other swashbuckler props.

See **Saving and Undoing** *for more discussion of handling random behavior in games*

Start of Chapter 7: Other Characters
Back to §7.4. Barter and Exchange
Onward to §7.6. Getting Started with Conversation

Example 130: ★ **Lanista 1** Very simple randomized combat in which characters hit one another for a randomized amount of damage.

Example 191: ★ **Red Cross** A DIAGNOSE command which allows the player to check on the health of someone.

Example 201: ★★ **Puff of Orange Smoke** A system in which every character has a body, which is left behind when the person dies; attempts to do something to the body are redirected to the person while the person is alive.

Example 293: ★★ **Lanista 2** Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an ATTACK IT WITH action is created to replace regular attacking. Also folds a new DIAGNOSE command into the system.

Example 113: ★★★ **Don Pedro's Revenge** Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

Example 127: ★★★ **Technological Terror** A ray gun which destroys objects, leaving their component parts

behind.

§7.6. Getting Started with Conversation

Traditionally, conversation is one of the most difficult things to program in interactive fiction, because of the number of factors affecting the outcome of everything the player does. While it's acceptable for >EXAMINE POT to produce the same response every time the player types it, it's a bit less acceptable for ASK JOE ABOUT HIS ADULTERY to make Joe react the same way every time.

Conversation implementations often need to keep track of a lot of information: what else is going on in the model world, what the character knows, what plot phase we've reached, what mood the character is in, what else we've recently been talking about, whether we've said the same thing before (and how many times); and so on. Later in this chapter we will look at ways to model character knowledge and mood.

Then, too, we have the problem of how the player communicates his conversational intentions to the story. Technology has not yet advanced to the point where a player can simply type in remarks in full natural English and have the character detect the significance, emotional tone, and subtext, if any: so we can't have RACHEL, THIS DESSERT TASTES LIKE FEET or WILL, LOOK! OUR SINISTER METAL FOES ARE APPROACHING! or BOSS, I WOULD BE DELIGHTED TO FILE ANOTHER TPB REPORT.

The challenge is to create an interface that is both easy for the player to use and expressive enough to be interesting. We will look at some of the common solutions in "Saying Complicated Things".

The examples in the following sections point out ways to approach common conversation problems. None of them will offer an adequate system if we want to write a very conversationally rich story, however. This is partly because a thorough conversation system requires quite a lot of code in its own right. It's also partly because there is no one right solution to the problem of conversation design. Different games will have quite different requirements. When making decisions about a new story we have planned, it may be useful to glance through the conversation extensions available for Inform: there are quite a few, offering a range of different interfaces. Even if none is exactly suited for our needs, they may suggest ways to solve particular implementation challenges.

At the other end of the scale, though, there are times when Inform's default implementation is too complicated for what we want to do: so we will start with ways to simplify conversation, before moving to all the exotic complexities.

Before we get into these details, though, we have a couple of examples that are literally about getting started with a conversation: **Mimicry** introduces the feature that we must greet other characters before beginning to speak to them; **The Gorge at George** corrects the player's attempts to use a TALK TO command where a different

mode of interaction is appropriate instead.

Start of Chapter 7: Other Characters

Back to §7.5. Combat and Death

Onward to §7.7. Saying Simple Things

Example 330: ★ **The Gorge at George** If the player tries to TALK TO a character, suggest alternative modes of conversation.

Example 110: ★★★ **Mimicry** People who must be greeted before conversation can begin.

§7.7. Saying Simple Things

There are times when even the commands ASK and TELL are overkill: sometimes the player doesn't have much information to offer, so TELL is never useful, for instance. If we don't want to make any distinction between modes of conversation, we can conflate the actions so that ASK LUCIUS ABOUT OLLIVANDER, TELL LUCIUS ABOUT OLLIVANDER and LUCIUS, OLLIVANDER all do the same thing: see **Sybil 1**.

If we are frequently permitting the player to say things like LUCIUS, OLLIVANDER as shorthand for "talk to Lucius about Ollivander", then we may also want to allow LUCIUS, OLLIVANDER? This makes the player character seem a bit slow (or at least Laconic), but it is an effective interface in some cases. The trick is that the question mark at the end of the command may prevent Inform from recognizing the keyword; should that problem arise, we may want to use Punctuation Removal to erase question marks from the player's command before attempting to interpret it.

Along the same lines, there are situations in conversation where similar commands do not correspond to the same actions within Inform; if we're careless about this, we may force the player to guess which vocabulary we want him to use, which is always vexing. Some cases to look out for:

Inform has actions for "saying yes" and "saying no". Sometimes this is useful, but sometimes we want YES and SAY YES TO FRED to do the same thing. **Sybil 2** shows how to roll these responses into one; **Proposal** expands on the idea to show more ways in which a player could reasonably answer a question put by another character.

Again, if we want ASK SYBIL ABOUT CAKE to do the same thing as SHOW CAKE TO SYBIL, we might use the technique in **Nameless** to make objects into valid topics of conversation, and to make ASK and SHOW behave the same way.

Finally, if we want to be able to ASK and TELL an inanimate object -- say, a computer -- about something, we may use the extension Inanimate Listeners to add this capability.

See **Remembering, Converting and Combining Actions** *for ways to redirect one conversation command to another conversation topic*

See **Varying What Is Read** *for a way of asking the player trivia questions that he can answer only on the next turn*

Start of Chapter 7: Other Characters

Back to §7.6. Getting Started with Conversation

Onward to §7.8. Saying Complicated Things

Example 91: ★ **Sybil 1** Direct all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

Example 93: ★★ **Sybil 2** Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

Example 171: ★★ **Proposal** Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

Example 297: ★★ **Nameless** ASKing someone about an object rather than about a topic.

---

§7.8. Saying Complicated Things

As we saw in the overview, there are challenges in choosing the commands with which the player will communicate to the story. Two common approaches are ASK/TELL conversation, where the player can ask or tell characters about keywords, as in ASK JILL ABOUT JACK or TELL FARMER ABOUT CHICKEN COOP, and menu-based conversation, where the player is offered a list of things to say and must pick one (often by number), as in

> 1) Ask Jill where Jack went.
> 2) Tell Jill that the chicken coop was robbed.

or, sometimes,

> 1) "Jill, have you seen your no-good layabout brother Jack anywhere?"
> 2) "Look, Farmer Jill, I think a fox got into the chickens."

The problem with ASK/TELL conversation is that it can feel undirected - if the player doesn't know which keywords to ask or tell about next, he gets stuck. It also doesn't always provide much sense of ongoing context or conversational flow, since the player can ask lots of unrelated questions and jump around a lot. What's more, sometimes the thing the player character asks isn't quite the question the player had in mind. If we type ASK JILL ABOUT JACK, Jill could wind up answering any of a number of questions - where Jack is, how old Jack is, whether Jack committed the recent murder, and so on. The player doesn't have much fine control over the conversation. Nonetheless, this is sometimes just what we want: **Farewell** implements a moderately sophisticated system along these lines, which keeps track of what the player has already said and allows him to review past conversation.

Menu-based conversation solves most of these problems: a branching tree of conversation choices maintains a consistent flow of discussion, it's hard for the player to run out of things to say, and the player always knows what his character is about to say. But there are compensating flaws. For one thing, a menu doesn't allow for many surprises. The player can see all the conversation the story has to offer by working methodically through all the menu branches. (This problem is sometimes referred to as the "lawnmower effect", since the process of seeing all the conversation is like the process of running a lawnmower over every inch of the lawn. It becomes a chore rather than an entertainment.) Menu systems can be long-winded to set up and therefore none are exemplified here, but several have been released as extensions for Inform.

Since about 2001, more and more IF has used a sort of compromise method: the player is allowed to ask or tell about keywords, but he's sometimes given prompts

about things to say that follow naturally on the conversation he was just having, as in

> You could ask where Jack is.

Moreover, when he asks about a topic where many comments are possible, he'll be allowed to clarify, either using a menu or through a disambiguation question such as

> >ask Jill about Jack
> Do you want to ask where Jack is, how old Jack is, or whether Jack committed the recent murder?

Sweeney implements one such hybrid type of conversation.

A third option is to take away almost all the player's expressiveness and give him just one command, TALK TO. The player can TALK TO characters whenever he wants, and the story will pick the most appropriate thing for him to talk about. This works best in works with few or simple puzzles and a fast-moving, constrained plot, where the player will keep having new things to talk about. **Cheese-makers** demonstrates this.

Finally, a few extreme games try to fake natural language understanding by looking for keywords in the player's input, rather than an exact grammar. This is perilous, because it is all too easy for the story to completely misunderstand what the player meant to type. Nonetheless, for the sake of example, see **Complimentary Peanuts**, in which the incomprehension is partly excused by the fact that the player is talking to someone a bit hard of hearing.

Start of Chapter 7: Other Characters
Back to §7.7. Saying Simple Things
Onward to §7.9. The Flow of Conversation
Example 281: ★★ **Farewell** People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.
Example 282: ★★ **Sweeney** A

conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

Example 168: ★★★ **Cheese-makers** Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

Example 376: ★★★ **Complimentary Peanuts** A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

---

§7.9. The Flow of Conversation

All this discussion of conversation commands and ways to model dialogue doesn't address the higher-level design issue: how do we approach writing this material so that it has a rhythm and flow? How do we know when we've created enough conversation? How can we avoid sounding hopelessly stilted when the nature of IF implementation requires us to break our text into small snippets?

While most authors develop their own approaches, there is some general advice that may help, especially for works that have a strong narrative progression.

It helps to have the plot of the story, with all its component scenes, planned in advance. That doesn't mean there can't be any changes later, but having a list of the different scenes can help us remember the different contexts in which information can appear. If we're using Inform's scenes feature, we may even want to restrict some dialogue to be available only during a given scene.

The next step is to go through scene by scene and create the "spine" of the scene. What *must* be said during this section? Is there anything the player can't leave without knowing? If the player isn't moving the scene forward fast enough, will the other character or characters volunteer information in order to keep the pace going?

It often helps to draft a transcript showing what we imagine as the ideal playthrough of the scene - writing straight through can create a natural flow of dialogue - before dividing the dialogue into pieces for implementation.

Once the scene is complete enough for the player to get through from beginning to end, we can start filling it out. At this point, it sometimes helps to play through the scene a number of times and add new dialogue elements as we think of things that our character might reasonably want to say. Sometimes these additions will turn out to be short tangents from the main flow of a very directed scene; sometimes they might be important branches that lead the scene to an entirely alternate outcome. The main thing is to make sure that, if the scene needs to hit certain points before ending, none of our branches keep the player from returning to the subject at hand.

Start of Chapter 7: Other Characters

Back to §7.8. Saying Complicated Things

Onward to §7.10. Character Emotion

---

§7.10. Character Emotion

In a complex story, characters may evolve strong feelings about the player. Often we want to hint at the character's feelings through gesture and tone of voice - little things woven into dialogue and action sequences that might otherwise be unchanged. **Ferragamo Again** demonstrates creating phrases to give all our characters different ways to express their irritation at the player.

Then again, sometimes a discussion might produce quite spectacular results if a character is in the wrong mood. **Being Peter** shows the bare bones of an implementation in which a character's attitude rulebook is consulted to determine what her response will be - allowing for arbitrarily complicated outcomes.

Start of Chapter 7: Other Characters

Back to §7.9. The Flow of Conversation

Onward to §7.11. Character Knowledge and Reasoning

Example 405: ★ **Being Peter** A set of rules determining the attitude a character will take when asked about certain topics.

Example 170: ★★ **Ferragamo Again** Using the same phrase

to produce different results with different characters.

§7.11. Character Knowledge and Reasoning

A character may be endowed with knowledge and even reasoning skills. Relations form quite a good way of keeping track of such problems: for instance, we can allow characters to be acquainted with one another with a relation such as

```
Lucy knows Lady Cardew.
```

Or we might keep track of more complicated attitudes between characters, as in **Murder on the Orient Express**, in which some characters suspect others of the crime.

Alternatively, we might have a list of salient facts that are important in our story. We might declare these as values, and then characters could know, learn, and forget entries as appropriate:

```
A fact is a kind of value. Some facts are defined by the Table of All
Known Facts.

Knowledge relates various people to various facts. The verb to know
(he knows, they know, he knew, it is known) implies the knowledge
relation.

Table of All Known Facts
fact summary
shoe-size "Lucy wears a size 9 shoe."
sunset-time "Sunset is at 8:22 PM this evening."

Lucy knows shoe-size.
Bob knows sunset-time and shoe-size.
```

Or again we might keep a whole database of information in a table: the characters in **Questionable Revolutions** know dates, countries, and a short description for each

of several rebellions and popular uprisings, while in **Queen of Sheba**, Solomon is able to answer who, what, where, when, and why questions about a range of topics. This kind of approach is most useful when the characters need to display a deep knowledge of a particular field. The facts stored in the Table of All Known Facts, above, are comparatively sparse, because there we are designing a story in which not all data about the world is equally valuable: Lucy doesn't know the shoe size of every person in the story, because for some reason it is only her own shoe size that matters. On the other hand, the Table of All Known Facts can store different kinds of information, whereas the revolutions table has no way of storing shoe sizes or sunset times. And **Murder on the Orient Express** works differently again, because it is storing knowledge that concerns people and things that already exist in the world model, rather than abstract ideas. Our way of modeling character knowledge, in other words, will depend quite a lot on what kind of knowledge it is.

The possibilities of character reasoning are similarly broad, but **The Problem of Edith** introduces one kind: the character has a concept of how different conversation topics relate to one another, so that when she is asked about a new keyword, she picks a response that makes the question most relevant to the conversation already in progress.

We end with a longer scenario, in which we track what the character knows about the player and the conversational state: in **Chronic Hinting Syndrome**, the main character guides conversation in the direction he intends it to go, with the player's sometimes-reluctant participation.

See **Obedient Characters** *for a character who needs to be taught how to perform actions before doing them*

See **Characters Following a Script** *for a programmable robot who can be given whole sequences of actions to perform*

Start of Chapter 7: Other Characters

Back to §7.10. Character Emotion

Onward to §7.12. Characters Following a Script

Example 241: **Murder on the Orient Express** A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but

there is physical evidence which may change their minds...

Example 235: **The Problem of Edith** A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before.

Example 278: **Questionable Revolutions** An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

Example 279: **The Queen of Sheba** Allowing the player to use question words, and using that information to modify the response given by the other character.

Example 341: **Chronic Hinting Syndrome** Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

---

§7.12. Characters Following a Script

So far we've seen characters who will answer questions whenever the player feels like asking, and characters who will use some reasoning procedure to direct the conversation. There is a third option, often useful in IF with a fast-paced narrative: the character follows a conversational script, making sure to cover a series of points before the scene ends.

There are more and less tedious ways to implement this kind of scene. The worst

case is one in which the player is not allowed to interrupt or ask any questions; he must merely wait until the character runs out of things to say. This can be useful and plausible in very small doses - say, two or three turns - but if the character has more information than that to impart, we may want to make the scene more interactive.

Pine 2 partly addresses this challenge: the character has a line of conversation that she wants to follow to its conclusion; we may ask questions along the way, but if we're silent, she'll take up the slack, and the scene won't end until she's done with what she has to say.

Another kind of script is a series of actions for the character to perform. **Robo** demonstrates a programmable robot that will observe what the player does, then try to emulate the actions later when switched into play-back mode. **Robo 2** extends this capacity to allow the robot to contain fifteen different scripts which the player can store, list, run, and erase.

Your Mother Doesn't Work Here offers a character with a list of tasks but whose plans can be interrupted by more urgent demands. This verges on not being a simple script any more: if we carry the idea to its natural conclusion, we get characters capable of planning scripts for themselves to accomplish their aims. This is conventionally called "goal-seeking".

See **Goal-Seeking Characters** *for characters that work out plans for themselves in order to accomplish various outcomes*

Start of Chapter 7: Other Characters
Back to §7.11. Character Knowledge and Reasoning
Onward to §7.13. Traveling Characters

Example 426: **Robo 1** A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

Example 438: **Your Mother Doesn't Work Here** Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

Example 161: ★★★ **Pine 2**
Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

Example 429: ★★★ **Robo 2**
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

---

§7.13. Traveling Characters

There are a number of ways we can make characters navigate our map. We might reasonably want them to approach and follow the player (as in **Van Helsing**); or to allow the player to follow characters who have left the room (as in **Actaeon**).

Characters who are less interested in the player will more likely follow their own courses around the available geography, however. A character may move randomly from room to room, as demonstrated in **Mistress of Animals**; he may follow a path that we have specifically written in advance, as **Odyssey** shows; or, most elegantly, he may use the "best route" calculation to find the best possible way to a given target room, as seen in **Latris Theon**.

This final method is arguably the neatest solution to character movement, allowing for characters to act in sophisticated ways; if we incorporate the Locksmith extension, other characters will even unlock and open doors that are in their way. The chief catch is that it should not be used too profligately with large numbers of characters, since on slow machines the processing power required to plan all their travel will make a noticeable difference to the running speed of the story.

All the same, the constraints are not so severe as to preclude having a moderate number of route-finding characters all wandering around at once. This does introduce a new problem, however: movement descriptions can become hard to follow if every turn produces long reams of reports such as

```
Joe enters the room from the south.
Lawrence opens the gate.
Lawrence departs to the west.
Lucy comes in from above.
Ted enters the room from the south.
```

> Bill departs to the west.

Patient Zero tackles this problem by calculating all of the character movement without printing any text; it then combines similar or related events into coherent paragraphs, as in

> Rhoda and Antony walk into the Post Office. Rhoda could have been rolling in chocolate and Antony looks as though dipped in french vanilla.

or

> Antony opens the iron gate. He goes through.

See **Doors, Staircases, and Bridges** *for some technical details of allowing other characters to interact with doors when they're in rooms that don't contain the player*

Start of Chapter 7: Other Characters

Back to §7.12. Characters Following a Script

Onward to §7.14. Obedient Characters

Example 77: **Mistress of Animals** A person who moves randomly between rooms of the map.

Example 39: **Van Helsing** A character who approaches the player, then follows him from room to room.

Example 274: **Odyssey** A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

Example 302: **Actaeon** A

FOLLOW command allowing the player to pursue a person who has just left the room.

Example 185: ★★★ **Latris Theon** A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

Example 402: ★★★★ **Patient Zero** People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

---

§7.14. Obedient Characters

Other characters can perform all the same activities that the player can; this does not always mean that they're willing to obey the player's instructions. By default, characters will refuse to obey commands of the form JULIA, WEST or ANTONY, TAKE THE PINCUSHION. Their objections can be overridden, however, and **The Hypnotist of Blois** implements a hypnotist who can make characters obedient at will.

In **For Demonstration Purposes**, the character is only capable of a few actions at the outset, but can be taught new ones if the player performs them first.

Often we want characters' obedience to be more selective. Just as the viewpoint character may be characterized in terms of what he will and will not do, so may others: **Generation X** demonstrates a character who will do what she's told, but who will comment unfavorably when the player asks for a nonsensical or repeated action, and who may eventually get fed up and leave.

Characters can be given moral objections to certain commands, as well: **Virtue** defines a few kinds of actions as bad, so that the character commanded will refuse to perform them.

Under Contract, more subtly, has the character object if the player's commands implicitly require any behavior he considers inappropriate: for instance, if the player commands him to put his pants in a container, he will work out that this requires the removal of the pants as a preliminary. If we want to implement a similar character, we may want to simply copy his unsuccessful attempt rule and the table of his retorts, then replace his banter with lines of our choosing.

The little example **Latin Lessons** allows us to make characters clever about vague commands: we can, for instance, write rules so that CLARK, EAT will have Clark sensibly pick something edible, rather than having the parser ask what we want Clark to eat.

Finally, **Northstar** demonstrates how we might make Inform understand commands of the form ASK JOSH TO TAKE INVENTORY or ORDER JOAN TO WEAR THE ARMOR.

See **Characters Following a Script** *for a programmable robot who can be given whole sequences of actions to perform*

Start of Chapter 7: Other Characters

Back to §7.13. Traveling Characters

Onward to §7.15. Goal-Seeking Characters

Example 184: **Virtue** Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

Example 186: **The Hypnotist of Blois** A hypnotist who can make people obedient and then set them free again.

Example 371: **Latin Lessons** Supplying missing nouns and second nouns for other characters besides the player.

Example 188: **Generation X** A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

Example 423: **Northstar** Making Inform understand ASK

JOSH TO TAKE INVENTORY as JOSH, TAKE INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

Example 239: ★★★ **For Demonstration Purposes** A character who learns new actions by watching the player performing them.

Example 205: ★★★★ **Under Contract** Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

---

§7.15. Goal-Seeking Characters

Goal-seeking characters are the most advanced IF life-form: they want to achieve specific outcomes, and they are able to work out plans of approach in order to bring these things about. They walk to rooms, open containers to search for things, use keys and tools, and ask leading questions in conversation.

A really advanced implementation of goal-seeking behavior is beyond the scope of our examples (though extensions exist that treat the problem more thoroughly). We can accomplish a surprising amount without heavy customization, though, if we keep in mind three points of technique:

First: it helps to think abstractly and to create broadly-defined actions as a first step to more specific tasks. For instance, a character's goal might be to eat some dinner. He'd be equally satisfied with spaghetti carbonara or with braised lamb shanks, but he needs to figure out which is available. So we might have our every turn rule (or whatever we're using to activate the character) say something like

```
Every turn when Clark is hungry:
    try Clark dining.
```

Dining would then be an action we've defined specially, which looks around Clark's

environment for suitable food; if it finds food, it issues a

```
try Clark eating the suitable food;
```

command; but if not, it sends Clark off to look for something likely. **The Man of Steel** demonstrates the use of this.

Second: though it doesn't actually contribute to the goal-seeking per se, lively reporting brings characters' generated behavior to life.

```
Clark eats a donut.
```

doesn't characterize Clark very much, even though the eating may be part of a subtle, intelligent plan to seduce Lois Lane. We'll do better if we replace a lot of the character reporting rules: to that end, see the example **The Man of Steel Excuses Himself**.

Third: goal-seeking characters notice when something is in the way of the action they want to perform. When that happens, they form a plan about how to remove the obstacle. We've already seen this kind of implementation on the player's behalf: the player will pick up items before eating them, say. We can use Before rules to do similar things for other characters, as in

```
Before Clark eating the wrapped candy:
    try Clark unwrapping the candy;
    if the candy is wrapped, stop the action.
```

Here we've set things up so that if Clark tries to eat the wrapped candy, he'll be interrupted by this other command; and if his unwrapping-the-candy attempt fails, he won't go on and eat the thing. **IQ Test** demonstrates a character who shows this kind of planning intelligence.

Because before-rules chain neatly, we can trigger whole plans of behavior if we have a sensible set, as in

```
Before someone entering a closed container: try the person asked
opening the noun.
Before someone opening a locked container: try the person asked
unlocking the noun.
Before someone unlocking a locked container: ...
```

We must exercise a little bit of care if it is possible for the chain of actions to produce

an endless loop - e.g., the character trying to take a key that is inside the transparent, locked box that it opens might repeatedly try to open the box, first unlocking the box, first taking the key, first opening the box, ... **Boston Cream** is a fully-worked scenario that deals with such a set of conundra.

See **Traveling Characters** *for characters who plan routes to locations and travel towards them*

See **Event Scheduling** *for characters who follow a pre-written schedule of activities*

See **Plot Management** *for having a central function direct all the characters in order to further the plot*

Start of Chapter 7: Other Characters
Back to §7.14. Obedient Characters
Onward to §7.16. Social Groups

Example 189: **IQ Test** Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

Example 203: **The Man of Steel** An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

Example 208: **The Man of Steel Excuses Himself** Elaborating the report rules to be more interesting than "Clark goes west."

Example 190: **Boston Cream** A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

§7.16. Social Groups

Crowds of characters introduce new challenges, because we often want to show them interacting with one another, or to describe individuals in less detail when a whole group is present.

Strictly Ballroom gives us a set of characters who pair off each turn, making sure to mention each one once, and leaving one unfortunate person behind as a wallflower: this exemplifies how we might use a behavioral rule not to dictate the behavior of each individual separately but rather to model a whole group together. **Happy Hour** does calculate movements for characters individually, but then collates the descriptions, creating a single paragraph to describe whatever group is currently in the room.

Characters can also have complicated attitudes to one another, and it can be helpful to use relations to track these. **Unthinkable Alliances** demonstrates the grouping of characters into alliance factions, while **The Abolition of Love** provides a host of relations to track love affairs, marriages, memberships in families, and mere mutual respect.

Emma combines these two effects: its characters move between social groups depending on how they feel about the others in their particular talking circle, and descriptions change depending on who is where in the room.

Lugubrious Pete's Delicatessen simulates a queue at a deli, in which the customers who most impress Pete get served first.

See **Traveling Characters** *for groups of characters who move around and have their movements collated into a joint description*

Start of Chapter 7: Other Characters
Back to §7.15. Goal-Seeking Characters
Onward to Chapter 8: Vehicles, Animals and Furniture: §8.1. Bicycles, Cars and Boats

Example 229: **Unthinkable Alliances** People are to be grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give

notice of quitting, and to become a lone wolf.

Example 231: ★ **The Abolition of Love** A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

Example 436: ★ **Lugubrious Pete's Delicatessen** In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

Example 176: ★★ **Strictly Ballroom** People who select partners for dance lessons each turn.

Example 353: ★★ **Emma** Social dynamics in which groups of people form and circulate during a party.

Example 356: ★★ **Happy Hour** Listing visible characters as a group, then giving some followup details in the same paragraph about specific ones.

Examples from Chapter 7: Other Characters

Start of this chapter

Chapter 8: Vehicles, Animals and Furniture

Indexes of the examples

You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer

and your local vicar here.

"Belfry"

The Belfry is a room. A bat is in the Belfry. The bell is in the Belfry. Some woodworm are in the Belfry. A man called William Snelson is in the Belfry. A woman called the sexton's wife is in the Belfry. A man called a bellringer is in the Belfry.

In the Belfry is a man called the vicar. The indefinite article of the vicar is "your local".

Test me with "look".

---

135 **ExampleClueless**

A murderer for the mystery is selected randomly at the beginning of the game.

WI

"When play begins" is the best point to initialize any aspects of the game that are meant to change between playings. For instance, in this scenario, we would randomly select one of the other characters to be guilty of murder:

"Clueless"

The murderer is a person that varies.

When play begins:
    now the murderer is a random person who is not the player.

The Billiards Room is a room. Colonel Mustard and Professor Plum are men in the Billiards Room. Miss Scarlet and Mrs White are women in the Billiards Room.

Instead of examining the murderer:
    say "[The noun] certainly looks fiendish!"

Test me with "x mustard / x plum / x scarlet / x white".

ExampleMeet Market

A case in which relations give characters multiple values of the same kind.

We have already seen that we can give things value properties -- a lamp has a brightness, for instance. Relations give us additional flexibility: since we may relate various things to various values, it is possible to describe a thing as having more than one value at the same time.

To demonstrate:

"Meet Market" by "K M and Eric Rossing"

Feature is a kind of value. The features are snub-nosed, gangly, comely, bright-eyed, and sulky.

Appearance relates various persons to various features. The verb to appear means the appearance relation.

Meet Market is a room.

Bob, Carol, Ted, and Alice are people in the Meet Market.

Bob appears snub-nosed and gangly.

Ted appears sulky and snub-nosed.

Carol appears comely and bright-eyed.

Alice appears bright-eyed and comely.

Yourself appears sulky and gangly.

Instead of looking:
    say "The snub-nosed ones: [list of people who appear snub-nosed][line break]";
    say "The gangly ones: [list of people who appear gangly][line break]";
    say "The comely ones: [list of people who appear comely][line break]";
    say "The bright-eyed ones: [list of people who appear bright-eyed][line break]";
    say "The sulky ones: [list of people who appear sulky][paragraph break]".

Test me with "look".

The same logic might be used to provide characters who have complex mood states: a person might be angry and sad, not merely one or the other -- feelings being what they are.

ExampleGopher-wood

Changing the name of a character in the middle of play, removing the article.

Suppose that we want a character who starts out with a general epithet ("the bearded man") but is later introduced to the player properly ("Japheth"). In that case, we want to be able to tell Inform to stop using an article once the character has been given a proper name. We can do this like so:

```
"Gopher-wood"

The Ark is a room. A bearded man is in the Ark.

Instead of examining the bearded man for the first time:
    now the printed name of the bearded man is
"Japheth";
    now the bearded man is proper-named;
    say "You peer at him a bit more closely and realize
that it's Japheth."
```

Finally, we need to tell Inform to understand the man's name, but only when he's been introduced. For this purpose, we borrow from the chapter on Understanding:

```
Understand "Japheth" as the bearded man when the
bearded man is proper-named.

Test me with "x japheth / x man / look / x japheth".
```

The peers of the English realm come in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince - and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

Almost all of this example is the flummery of pomp and circumstance: only the first two paragraphs really do anything.

"Peers" by Elizabeth II R

A title is a kind of value. The titles are Baron, Viscount, Earl, Marquess, Duke and Prince.

A peer is a kind of man. A peer has a title. A peer is usually a Baron. Before printing the name of a peer, say "[title] ". Understand the title property as describing a peer.

The House of Lords is a room. Maltravers, Pollifax, Omnium and St Vincent are peers in the House of Lords. Omnium is a Duke. St Vincent is an Earl.

Ennobling is an action applying to one thing and one title.

Check ennobling:
    if the noun is the player, say "The Sovereign is the fountain of honour, and may not be ennobled." instead;
    if the noun is not a peer, say "Commoners should remain so." instead;
    if the title of the noun is the title understood, say "But that is his title already." instead;

if the title of the noun is greater than the title understood, say "As he is already of the rank of [title of the noun], any such letters patent are liable to be deemed invalid, following the precedent of the Buckhurst Peerage Case (1876). Best not." instead.

Carry out ennobling:
    now the title of the noun is the title understood.

Report ennobling:
    say "'Whereas Our Parliament for arduous and urgent affairs concerning Us the state and defence of Our United Kingdom and the Church is now met at Our City of Westminster We strictly enjoining Command you upon the faith and allegiance by which you are bound to Us that the weightness of the said affairs and imminent perils considered (waiving all excuses) you be at the said day and place personally present with Us and with the said Prelates Great Men and Peers to treat and give your counsel upon the affairs aforesaid And this as you regard Us and Our honour and the safety and defence of the said Kingdom and Church and dispatch of the said affairs in nowise do you omit Witness Ourself at Westminster the Fifth day of November in the 43rd year of Our Reign,' you say, with unpunctuated serenity. The new [noun] bows stiffly."

Understand "dub [someone] a/an [title]" as ennobling.

Test me with "dub st vincent a baron / dub maltravers a marquess / look / examine marquess".

ExampleA Humble Wayside Flower

Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

First we define the relationships we choose to acknowledge:

"A Humble Wayside Flower"

Marriage relates one person to another (called the spouse). The verb to be married to means the marriage relation.

Fatherhood relates one person (called father) to various people. The verb to engender means the fatherhood relation.

For brevity, we will ignore the existence of mothers. It is a sad world.

Siblinghood relates a person (called A) to a person (called B) when a person who engenders A engenders B. The verb to be sibling to means the siblinghood relation.

Family relates a person (called A) to a person (called B) when A is married to B or A engenders B or B engenders A or A is sibling to B. The verb to be related to means the family relation.

A person can be known or unknown. After printing the name of an unknown person (called the alien):
    if a known person (called the contact) is related to the alien:
        say " ([relation between alien and contact] of [the contact])";
        now the alien is known;
        rule succeeds.

To say relation between (first party - a person) and

```
(second party - a person):
    if the first party is married to the second party:
        if the first party is female, say "wife";
        otherwise say "husband";
        rule succeeds;
    if the first party is sibling to the second party:
        if the first party is female, say "sister";
        otherwise say "brother";
        rule succeeds;
    if the first party engenders the second party:
        say "father";
        rule succeeds;
    if the second party is the father of the first party:
        if the first party is female, say "daughter";
        otherwise say "son";
        rule succeeds.
```

```
Pere Blanchard's Hut is a room. Percival Blakeney is a
known man in the Hut. Marguerite is a woman in the
Hut. Percival is married to Marguerite. Outside from the
Hut is the Garden. Louise is a woman in the Garden.
The Road to Paris is west of the Garden. Armand St
Just is a man in the Road. Louise is married to Armand.
Monsieur St Just is a man. He engenders Armand and
Marguerite.
```

```
Test me with "out / west / east / west".
```

Monsieur St Just never appears on the scene in this piece, but if we did put him somewhere the player could find him, he, too, would be properly introduced.

Example **Annoyotron Jr**

A child who after a certain period in the car starts asking annoying questions.


```
"Annoyotron Jr"
```

```
The Minivan is a room. The Open Road is outside from
the Minivan. Pete is a man in the Minivan. "Pete [if the
player has been in the Minivan for 3 turns]is starting to
look bored[otherwise]is playing with his travel activity
book[end if]."


Every turn:
    if the player has been in the Minivan for 5 turns, say
"'Are we there [if saying no]now?'[otherwise]yet?' asks
Pete.[end if]"


Instead of saying no:
    say "'Oh,' says Pete. There is a blessed, momentary
silence."


Instead of going to the Open Road:
    say "You leap to your death.";
    end the story.


Test me with "z / z / look / g / g / g / no / z / z / z / no / z /
out".
```

Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

```
"Camp Bethel"


Camp Bethel Kitchen is a room.
```

One use for text alternatives is to change the description of a room after

first visiting. We've already seen, in the example "Slightly Wrong", how to do this with "[if visited] ... [otherwise] ... [end if]". But since the first description is printed once and the second description on all subsequent occasions, we could equally well write

```
The description of Camp Kitchen is "[one of]You've
never been into the kitchen before, though you've spent
many an hour in the dining lodge. The place is larger
than you would have expected, and it has none of the
fake rustic touches of the rest of the camp[or]A tidy,
efficient industrial kitchen, without any of the kitsch
rusticity found elsewhere[stopping]."
```

We might also want to liven up the behavior of people and animals, who are probably not doing the exact same thing every time we glance in their direction. There are more complex techniques for modeling the behavior of characters, as we will see in the chapters on Advanced Actions and Activities; but if we just want some textual variety, we might write something like:

```
Jeremy is a man in the Camp Bethel Kitchen. "Jeremy
stands at his station, [one of]peeling white
onions[or]briskly dicing onions[or]chopping
celery[or]peeling carrots[or]tying fresh herbs together
with string[or]putting all the vegetables into a large stock
pot[or]watching over his boiling vegetable
stock[cycling]."
```

And since (textual variation or not) we do want the player to be able to see all these objects:

```
Jeremy carries white onions, celery, carrots, and herbs.
Jeremy's station is scenery in the kitchen. It is a
supporter.
```

Jeremy is following a sequence of actions to do an implied task (still somewhat robotically, but it will do for now). Animals might be a bit more capricious, though:

```
Fluffy is an animal in the Camp Bethel Kitchen. "[one
of]Fluffy is chasing its tail[or]Fluffy is staring out the
```

```
window[or]Fluffy is rubbing itself against your leg[purely
at random]."
```

```
A housefly is an animal in the Camp Bethel Kitchen. "A
large housefly [one of]lands on a countertop[or]flies
around noisily[or]circles Jeremy's chef hat[at random]."
```

The housefly's description is merely "at random" rather than "purely at random" because we want to show it doing a different thing each turn, whereas Fluffy could plausibly stare out the window for five turns in a row.

There are more complex ways to change and override the initial descriptions of people and things; if text alternatives do not get us far enough, we can turn to the "rule for writing a paragraph about," documented in the Activities chapter.

Another frequent use of text alternatives is to give characters a bit of variety in things they're likely to say many times in the course of a game:

```
Instead of telling Jeremy about something:
    say "Jeremy looks [one
of]surprised[or]intrigued[or]nonplussed[at random]. '[one
of]You don't say[or]That's very interesting[or]Do go
on[or]I wish I'd known that sooner[at random]!'".
```

Or, with somewhat more complexity:

```
Instead of asking Jeremy about something:
    say "'[one of]Sorry,[or]I'm afraid[or]Hm,[at random]
[one of]I don't know much about that[or]you've got me
there[or]I haven't the faintest[at random],' Jeremy [one
of]drawls[or]replies[or]comments[or]exclaims[at
random]";
    say "[one of][or] huskily[or] throatily[or] silkily[or] in a
deep manly voice[as decreasingly likely outcomes]."
```

Notice that, in that last line, our first option is entirely blank. If we put nothing as an element of the text alternatives list, this means that printing nothing at all is a viable alternative. In fact, we've made this the most common probability out of the decreasingly likely outcomes, so that five times in fifteen, or a third of the time the text is run, there will be no modifier printed at all.

Test me with "look / g / g / g / ask Jeremy about his feelings for me / ask jeremy about his amnesia / tell Jeremy about my unborn child".

As this example (alas) reveals, text alternatives will not go all the way toward making our characters into compelling conversationalists; we will have to wait until we know more about Actions. But at least we have abolished the default responses, and given Jeremy a touch of personality, however witless.

ExampleLean and Hungry

A thief who will identify and take any valuable thing lying around that he is able to touch.

"Lean and Hungry"

Substance is a kind of value. The substances are silver, gold, and lead. Everything has a substance. A thing is usually lead.

Definition: a thing is valuable if it is not lead.

The Limestone Cave is a room. "Not very big, and it doesn't go back far, but you'd hoped to find some shelter here. Outside it is raining exceptionally hard."

The sinister gentleman is a man in the Cave. "Leaning against the wall is a sinister gentleman in a threadbare waistcoat." The description is "He looks as though he might once have been quite well off."

```
After examining the gentleman:
    say "[The noun] smiles back at you in an unnerving
fashion."
```

Now we make the rule that governs the gentleman's behavior. Here we're going to invoke the rules that allow characters besides the player to do actions. More about this can be found in the Advanced Actions chapter:

```
Every turn:
    if the sinister gentleman can touch something
valuable (called the treasure) which is not carried by a
person:
        try the gentleman taking the treasure.


Report the gentleman taking something:
        say "[The gentleman] slyly acquires [the noun] and
tucks it into his pocket." instead.
```

That "not carried by a person" prevents the gentleman from stealing from the player (or, less plausibly, from himself). If we did want him to pick pockets, we could just have said "which is not carried by the gentleman".

```
The player is carrying a coin, a bust of Abraham Lincoln,
a bottle of sherry, a small pistol, and a wad of
Confederate cash. The coin is gold. The pistol is silver.


A locket is in the Cave. The locket is gold. "A broken
gold locket lies on the ground, a token of your late
Mama."


Test me with "get locket / i / drop locket / z / drop cash /
z / x gentleman / drop all / z / z / look".
```

A game of foosball which relies heavily on every-turn rules.

Suppose we want a game of foosball in which our opponent acts every turn, but does different things depending on where the ball currently lies. We can put together a sequence of every-turn rules to account for this, as follows:

```
"Text Foosball"

Use scoring.

The Lounge is a room. "The Lounge is appointed with
everything necessary to rest and relaxation: a vending
machine, a potted palm, a stack of Entertainment
Weekly issues from 1993, and -- your pride and joy -- a
foosball game."

The foosball game is scenery in the Lounge.
Understand "table" or "football" or "foozball" or "fussball"
or "soccer" as the foosball game. The game is a
supporter. On the game is a small white ball. The ball
can be still, approaching, receding, or unreachable. The
description of the ball is "Currently [small white ball
condition]."

After printing the name of the small white ball, say "
([small white ball condition])".

When play begins:
    now left hand status line is "You: [score]";
    now right hand status line is "Joey: [Joey's score]".
```

Some tiny men on sticks are part of the game. Understand "handles" as the tiny men. The description is "Okay, a couple of the tiny men have had their feet broken off, and the table surface itself is a bit warped, and the ball resembles a quail egg in respect of shape and color. This makes for a game of unusual randomness, but skill is overrated."

Instead of attacking or pulling or pushing the game when the ball is unreachable:
    say "You give the table a good shove, and the ball moves ever-so-slightly.";
    now the ball is still.

Instead of taking the white ball:
    say "You'd forfeit the game if you did that."

Instead of turning the tiny men when the ball is unreachable:
    say "The ball has somehow gotten to a mystical point on the table where it cannot be reached, no matter what. Close inspection reveals that this point has been marked in chalk with a tiny X. Not that that does any good."

Instead of turning the tiny men when the ball is approaching:
    if a random chance of 2 in 3 succeeds:
        if a random chance of 1 in 2 succeeds, now the ball is receding;
        otherwise now the ball is still;
        say "[if the ball is still]Thunk. [otherwise]Thwack! [end if]You keep the ball from reaching its goal! Now it is [small white ball condition].";
    otherwise:
        let Joey score.

To let Joey score:
    now the ball is still;

now Joey's score is Joey's score + 1;
say "The ball rolls neatly into your goal, despite your efforts. ";
if Joey's score < score, say "You put the ball back in the center with a snap. No reason to worry yet; you're still ahead. Joey looks determined, though.";
otherwise say "After allowing a moment or two for Joey's gloating to pass, you replace it at the center."

Instead of turning the tiny men when a random chance of 1 in 13 succeeds:
if the ball is unreachable, continue the action;
now the ball is unreachable;
say "You hit the ball off-center and it rolls sluggishly into a little dip in the surface of the foosball table. ";
if Joey's score > 7, say "'You did that on purpose!' Joey exclaims indignantly.";
otherwise say "You and Joey exchange glances. This is never good."

Instead of turning the tiny men:
say "You madly rotate the tiny men on sticks! ";
if a random chance of 1 in 2 succeeds:
say "Hoorah! You hit the ball!";
now the ball is receding;
otherwise:
say "Somehow you fail to bring your monopodal player into contact with the ball."

Joey is a man in the Lounge. "Joey is hunkered over the foosball handles on his side of the table." Joey can be active or inactive.

Joey's score is a number that varies.

Every turn: now Joey is active.

```
Every turn when the ball is approaching and Joey is
active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    now Joey is inactive;
    let Joey score;
    rule succeeds.


Every turn when the ball is unreachable and Joey is
active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    now Joey is inactive;
    say "Joey glares angrily at the stuck ball."


Every turn when the ball is receding and Joey is active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    if the ball is unreachable, make no decision;
    now Joey is inactive;
    if a random chance of 1 in 2 succeeds:
        if a random chance of 1 in 2 succeeds, now the ball
is still;
        otherwise now the ball is approaching;
        say "Joey connects with your shot. Now the ball is
[small white ball condition]!";
    otherwise:
        now the ball is still;
        say "Joey tries to block, but misses! Back it goes in
the center, where it is [small white ball condition].";
        increment the score.


Every turn when the ball is still and Joey is active:
    let total be Joey's score + score;
    if total > 9, make no decision;
    if the ball is unreachable, make no decision;
    now Joey is inactive;
    if a random chance of 1 in 2 succeeds:
        now the ball is approaching;
        say "Joey hits the ball solidly down towards your
goal. Now it is [small white ball condition].";
    otherwise:
        say "Joey fails to hit the ball in your direction. It
remains [small white ball condition]."
```

```
Every turn:
    let total be Joey's score + score;
    if total > 9:
        if Joey's score > score, end the story saying "Rats!
Joey wins!";
        if Joey's score < score, end the story finally saying
"Victory is yours!";
        if Joey's score is score, end the story saying "A
perfect tie."
```

ExampleZodiac

Several variations on "doing something other than...", demonstrating different degrees of restriction.

Notice that the following two scenarios do not have the same effect:

```
"Zodiac"

The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.

The player carries a can of mace and a roll of duct tape.

Instead of doing something other than examining with
the Capricorn Killer:
    say "You wouldn't dare!"

Test me with "x me / x killer / touch killer / smell mace".
```

...will prevent the player from doing anything else to the killer, but allow him free range of action with other objects. By contrast, the following will

prevent him doing anything other than examining to any item:

```
"Zodiac"

The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.

The player carries a can of mace and a roll of duct tape.

Instead of doing something other than examining
something in the presence of the Capricorn Killer:
    say "You dare not attempt it!"

Test me with "x me / x killer / touch killer / smell mace".
```

But notice that because we specified "examining something", actions that take no object (like sing) are still not affected.

The most strict statement would be

```
"Zodiac"

The Secluded Alley is a room. The Capricorn Killer is a
man in the Secluded Alley.

The player carries a can of mace and a roll of duct tape.

Instead of doing something other than looking or
examining in the presence of the Capricorn Killer:
```

say "You dare not attempt it!"

Test me with "x me / x killer / touch killer / smell mace".

Because we left "something" out, now the pattern does not have to match an activity with an object; it will match any activity at all. Singing too will be deemed too risky in the presence of our sinister foe. But we do need to make an exception for "look", or else we won't get even the room description.

ExamplePine 1

Pine: Using a scene to watch for the solution of a puzzle, however arrived-at by the player.

Because scene rules are checked every turn, they can be useful for designing puzzles which have multiple solutions. Instead of deciding the puzzle is "solved" when the player does a certain action, we set up a scene that checks to see whether the player has achieved a certain outcome -- however he accomplished it.

For instance, in this scenario, we're waiting for Sleeping Beauty to wake up, and it doesn't much matter how...

"Pine"

A person can be asleep or awake. A person can be active or passive.

The Spinning Tower is a room. "A remote corner of the old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning Tower. "[if asleep]Sleeping Beauty lies here, oblivious to your presence[otherwise]Sleeping Beauty stands beside

you, looking a little confused[end if]." The description is "She is even more magnificent than the rumors suggested." Understand "woman" or "girl" or "princess" or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play begins. Discovery ends when Sleeping Beauty is awake. Marriage Proposal is a scene. Marriage Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you can hear the other sounds of stirring and movement as the spell of centuries is broken."

Instead of waking an awake person: say "Redundant."

Instead of waking an asleep person: say "Yes, but how?"

Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much for that true love's kiss nonsense."

Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"

Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].

[The second noun] wakes, shuddering. 'Agh! I had a terrible dream about drowning and then-- Hey!'"

The player carries a jug of water. Understand "pour [something] on [something]" or "splash [something] at/on [something]" as throwing it at.

Test me with "x beauty / wake beauty / pour water on beauty".

---

187 **Example** **Police State** *WI*

Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

"Police State"

Seventh Avenue is a room. "The bars are all closed now, and there aren't any good clubs to go to, so you're on your own for open-air entertainment."

Persuasion rule for asking the policeman to try doing something: persuasion fails.

Persuasion rule for asking someone to try doing something: persuasion succeeds.

Note that the policeman will never get to the second persuasion rule, so he will always refuse to do the player's nefarious bidding.

```
Charles, Thomas, and Larry are men in Seventh
Avenue. Patricia is a woman in Seventh Avenue.
```

And here's an unnecessary aesthetic touch from a later chapter, which will round up the descriptions of your friends into a single paragraph:

```
Rule for writing a paragraph about someone who is not
the policeman:
    let X be the number of visible people who are not the
policeman;
    say "It's just [X in words] of you now: [a list of visible
people who are not the policeman]. But it sure has been
a rip-roaring evening."


The policeman is a man in Seventh Avenue. "A
policeman with a very guarded expression is watching
you."


Singing is an action applying to nothing. Understand
"sing" as singing.


Report singing:
    say "A little the worse for wear, you sing."


Smelling a person is disorderly conduct. Tasting a
person is disorderly conduct. Jumping is disorderly
conduct. Singing is disorderly conduct.


Instead of someone trying disorderly conduct in the
presence of the policeman:
    say "The policeman arrests [the person asked]!";
    now the person asked is nowhere;
    the rule succeeds.
```

```
Instead of disorderly conduct in the presence of the
policeman:
    end the story saying "The policeman arrests you!"
```

```
Test me with "charles, look / charles, jump / look /
policeman, sing / thomas, taste policeman / patricia,
sing / look / jump".
```

Notice the difference between the two rules about disorderly conduct: the one for other people says 'the rule succeeds' to make sure that the action is counted as a success and not (as normally happens with instead rules) a failure. Most of the time we don't care whether actions are judged successes or failures, but it matters here, because if we type CHARLES, JUMP and the result fails, then text such as 'Charles is unable to.' will be printed - which would get in the way. So we declare the action a success.

ExampleSearch and Seizure

A smuggler who has items, some of which are hidden.

Suppose we want a character who carries hidden objects, but only while he is wearing his jacket. If we deprive him of this, his other possessions become known. Furthermore, if we ever search him, his possessions also become known, and can thereafter be mentioned by us.

```
"Search and Seizure"
```

```
Size is a kind of value. The sizes are small, medium,
and large. A thing has a size. A thing is usually small.
```

```
A thing can be licit or contraband. A thing is usually licit.
```

```
A thing can be discovered or secret. A thing is usually
secret.
```

Now, we want the character to be able to hide small things if he has some sort of concealing garment on. We also want to be able to see anything that the player has already found once, perhaps by using the >SEARCH PERSON command. So:

```
Rule for deciding the concealed possessions of
someone (called the suspect):
    if the particular possession is discovered, no;
    if the suspect wears something and the particular
possession is small, yes;
    otherwise no.
```

The following rule, borrowed from a later chapter, assures that any items that are ever mentioned to the player will be treated as known from then on:

```
Before printing the name of something (called
discovery): now the discovery is discovered.
```

```
The Customs House is a room. The smuggler is a man
in the Customs House. The smuggler wears a leather
jacket. He carries a bottle of Altairan rum, some
raspberries, and a laser pistol. The pistol is large. The
jacket is large. The rum and the raspberries are
contraband. The description of the smuggler is "He has
a bestubbled chin and a sinister eye."
```

```
The tourist is a woman in the customs house. The
description is "The type who walks off home with a
dozen contraband items in her pocket not because she
means to steal things but because she's too stupid to
understand that the law applies to herself." She wears a
tight-fitting dress. The dress is large. The tourist carries
a grapevine and an archaeological artifact. The
grapevine is large and contraband. The artifact is
contraband.
```

```
Report examining someone: say "[The noun] is [if the
noun is wearing something]wearing [a list of
```

```
unconcealed things worn by the noun] and [end
if]carrying [a list of unconcealed things carried by the
noun]."


Report examining someone who is concealing
something contraband:
    say "[The noun] looks nervous. You can just tell."


Report examining someone who is carrying an
unconcealed contraband thing:
    say "Your eye goes at once to [the list of unconcealed
contraband things carried by the noun] which [the noun]
is unable to hide";
    if an unconcealed licit thing is had by the noun, say ",
though [if the noun is female]s[end if]he also has [a list
of unconcealed licit things had by the noun]";
    say "." instead.
```

Notice that we can talk about what the smuggler wears, what he carries, and what he "has": things the smuggler has can be either worn or carried, so the phrase is useful if we don't care to make that distinction.

```
Instead of searching someone:
    say "[The noun] is revealed to be carrying [a list of
things carried by the noun]."


Instead of confiscating the dress:
    say "You are not allowed to perform strip-searches in
the public customs area."


Understand "confiscate [something]" as confiscating.


Confiscating is an action applying to one thing.
```

```
Check confiscating:
    unless the noun is had by someone who is not the
player:
        say "You can only confiscate other people's
possessions." instead.

Carry out confiscating:
    now the noun is carried by the player.

Report confiscating:
    say "Through the authority vested in you by the
power of the Sovereign of Centauri Proxima, you make
[the noun] your own."

Test me with "x smuggler / search smuggler / x
smuggler / confiscate jacket / x smuggler / confiscate
rum / confiscate pistol / x smuggler / confiscate
raspberries / x smuggler".

Test more with "x tourist / confiscate dress / confiscate
grapevine / x tourist / search tourist / confiscate artifact /
x tourist".
```

ExampleNoisemaking *WI*

Creating a stage after the report stage of an action, during which other characters may observe and react.

Suppose the current sequence of action handling is not quite enough for us: we'd also like to have a stage after reporting, where other characters can react to the player character's behavior after it has already happened and been reported on screen. Having such a stage is unlike using "after", because after occurs before reports and prevents them from being printed. So, for instance, we could allow the player to do any of a range of different actions that make loud noises, and have a nervous bird that reacts to all of them by flying away afterward.

To do this, we can add a new rule into the specific action-processing rules. (For a list of these, see the Rules index.) Moving rules around and adding new ones requires syntax that we will learn in the chapter on Rulebooks, but the present example is fairly straightforward:

```
"Noisemaking"

Section 1 - Procedure

The other-player response rule is listed after the report
stage rule in the specific action-processing rules.

This is the other-player response rule:
    follow the observation rules.

The observation rules is a rulebook.

Section 2 - Scenario

Country Lane is a room. West of Country Lane is
Outside the Farmhouse. East of Country Lane is Village
Center. North of Country Lane is Open Field.

The player carries a drum.

The black crow is an animal in Country Lane.
```

Singing is an action applying to nothing. Understand "sing" as singing.

Report singing:
    say "You hum a little ditty."

Singing is a loud action.

Attacking the drum is a loud action.

The block attacking rule is not listed in any rulebook.

Report attacking something:
    say "THWACK!"

An observation rule for loud action in the presence of the black crow:
    let N be a random adjacent room;
    if N is a room, move the black crow to N;
    say "The crow, startled, flies off to [N]."

Test me with "sing / g / n / hit drum".

Example **Revenge of the Fussy Table**
A small game about resentful furniture and inconvenient objects.

This example does include a number of features that we haven't met yet, particularly rulebooks. Nonetheless, the basic idea should be relatively

clear. Our premise is that the player is faced with a series of disgruntled furnishings, none of which want to be responsible for supporting the cold, hot, drippy, or spiky objects lying around the room.

"Revenge of the Fussy Table"

Use scoring.

Section 1 - The Supporters

The Dining Room is a room. "A room of handsome proportions, with an octagonal dining table on a fine parquet floor. The original set of chairs has been taken away, leaving only a red velvet dining chair and a blue suede armchair."

The fussy table is a supporter in the Dining Room. It is scenery. The description is "It's hypochondriac: the result of a sojourn in a superior antique shop, where it picked up the scent of almond oil, words like provenance, and a secretive shame about being resurfaced.

Chairs don't get along with it.

In its mellowest moods, after a really good dusting, it will mention, casually, a cabinet it once knew, which was twelve feet tall, came from the Winter Palace, and had an asking price of $350K."

A chair is a kind of supporter with carrying capacity 1. A chair is enterable. A chair is usually scenery.

Understand "chair" as a chair.

The blue suede armchair and the red velvet dining chair are chairs in the Dining Room. The description of the blue suede is "Quiet, mellow, and with a lingering aroma of clove cigarettes, and possibly something less legal. It doesn't sit up straight, and doesn't let anyone else do so either." The red velvet has the description "In temperament, in bearing, in everything it is the opposite of the blue suede chair. It is concerned for its duty all the time, has a rigidly straight back, and pokes diners in the spine when it suspects them of overdoing things."

The carrying capacity of the player is 2.

Section 2 - Sources of Unpleasantness

A thing can be drippy or dry.

Heat is a kind of value. The heats are luke-warm, cold, and scalding. Everything has a heat. Understand the heat property as describing a thing.

A thing can be spiky or smooth. The cleated left shoe is a wearable spiky thing. It is worn by the player. The cleated right shoe is a wearable spiky thing. It is worn.

Definition: a thing is unpleasant if it is scalding or it is cold or it is spiky or it is drippy.

Definition: a supporter is occupied if something is on it.

Definition: something is contented if it is not concerned.

Definition: a chair is concerned if something unpleasant is on it.

Definition: the table is concerned if something drippy is on it or something scalding is on it or something spiky is on it.

Definition: the ceramic tile is concerned if something spiky is on it.

Definition: a room is concerned if something drippy is in it or something scalding is in it or something spiky is in it or someone which is in it is wearing a spiky thing.

Before printing the name of a drippy thing: say "drippy ".
Before printing the name of a cold thing: say "cold ".
Before printing the name of a scalding thing: say "scalding ".

Section 3 - What the Player Can Do About It

After putting a cold thing on a scalding thing:
    say "[The noun] meets [the second noun]; both shriek in pain. But the necessary heat transfer occurs.";
    now the heat of the noun is luke-warm;
    now the heat of the second noun is luke-warm.

After putting a scalding thing on a cold thing:
    say "[The noun] meets [the second noun]; both shriek in pain. But the necessary heat transfer occurs.";
    now the heat of the noun is luke-warm;
    now the heat of the second noun is luke-warm.

After taking a scalding thing:
    say "'Right,' you say. 'I'll just hold onto [the noun], shall I? Because that won't be in the least uncomfortable for me.'"

After taking a drippy thing:
    say "With a sense of martyrdom, you retrieve [the noun], which dribbles water down your cuffs."

After entering a chair when the player is wearing a spiky thing (called the clog):
    say "You sit on [the noun], lifting [the list of spiky things worn by the player] so that the parquet is no longer affronted."

Section 4 - In Which the Furnishings Complain

An every turn rule:
    follow the complaint rules.

The complaint rules is a rulebook.

A complaint rule:
    if something (called the offending item) on the table is drippy:
        say "'Help! Get me a coaster!' screams the table[if the table is visible], its veneer squirming under [the offending item][otherwise] from the Dining Room[end

```
if].";
        rule succeeds;
    if something (called the offending item) on the red
chair is drippy:
        say "'Oh dear,' murmurs the red chair, as [the
offending item] drips into its velvety seat. 'Oh dear, I will
have a damp spot. This is so very -- what will people
think?'";
        rule succeeds;
    if something (called the offending item) on the visible
armchair is drippy:
        say "[The offending item] visibly begins degrading
the suede where it sits. The armchair is tactfully silent.";
        rule succeeds;
    if a drippy thing (called the offending item) is in the
location and the player is in the Dining Room:
        say "'Cripes,' says the parquet. 'No one mind me at
all. Just leave that [offending item] right here. You know
I'm the most valuable thing in the room?'";
        rule succeeds.



A complaint rule:
    if a cold dry thing (called the offending item) is on the
table:
        say "The table shivers under [the offending item].";
        rule succeeds.



A complaint rule:
    if something (called the offending item) on the table is
scalding:
        say "'Hey!' protests the table, practically smoking at
[the offending item]. 'Heard of a trivet?'";
        rule succeeds;
    if something (called the offending item) on the
armchair is scalding:
        say "The armchair doesn't complain about [the
offending item]; it just begins, quietly, to give off warm
sweet clouds of blue steam, as though its inner opium-
nature had been released.";
        rule succeeds;
    if something (called the offending item) in the Dining
Room is scalding:
        say "'Yes indeed,' [if the location is the Dining
Room]says[otherwise]bellows[end if] the parquet. 'That
[offending item] won't leave any sort of mark what-so-
ever.'";
```

rule succeeds.

A complaint rule:
if something (called the offending item) on the table is spiky:
say "The table holds very very very still lest [the offending item] mar its finish. But its resentment is palpable.";
rule succeeds;
if something (called the offending item) on the velvet chair is spiky:
say "[The offending item] stabs the velvet of the red velvet chair, which draws itself up even more sharply in a pose of dutiful martyrdom.";
rule succeeds;
if something (called the offending item) which is spiky is on the ceramic tile:
say "'[The offending item] itches!' cries the ceramic tile.";
rule succeeds;
if something (called the offending item) in the location is spiky:
say "'[The offending item] should not be on a parquet floor,' sings the parquet floor sadly.";
rule succeeds;
if someone (called the offending person) in the location is wearing something spiky and the location is the Dining Room:
say "The parquet floor mutters about the things people wear to walk on parquet floors these days.";
rule succeeds.

A complaint rule:
if the table supports something which supports a drippy thing (called the offending item),
say "'[The offending item] could still theoretically dribble down here,' says the table nonchalantly. 'It's possible.'";
if the velvet chair supports a concerned thing (called the friend),
say "'[The friend] seems a little worried,' remarks the velvet chair helpfully."

A complaint rule:

```
    if the concerned table can see a supporter (called the
object of envy) which is not concerned:
        say "The table looks jealously at [the object of
envy]. 'Some people have such an easy life.'";
        rule succeeds.
```

```
Instead of going to an adjacent room when the player is
in the Dining Room:
    say "'You're not leaving!?' whimpers the table at
once."
```

Moreover, every time the player gets close to resolving this issue, his unhelpful companion Alison brings in something else inconvenient. We can use the counting of contented supporters to decide when the player is getting close to winning and it's time for her to bring something else...

```
Section 5 - Source of Further Complications
```

```
The Tidy Kitchen is south of the Dining Room. Alison is
a woman in the Tidy Kitchen. The kettle is a scalding
thing. The ceramic tile is a cold portable supporter. It
has carrying capacity 1. The platter is a scalding
portable supporter. It has carrying capacity 1. The glass
of ice water is a cold, drippy thing.
```

```
Instead of drinking the glass of ice water, say "It's not
yours, it's for Alison, once she finishes cooking."
```

```
Alison carries the kettle, the tile, the platter, and the
glass.
```

```
An every turn rule:
    follow the behavior rules.
```

The behavior rules is a rulebook.

A behavior rule:
    if ice water is carried by Alison and the player is in the Dining Room and almost all of the supporters are not concerned:
        let the next victim be a random fixed in place contented supporter;
        move the ice water to the next victim;
        say "Humming to herself, Alison brings in [an ice water] and sets it down smack on [the next victim], then goes back out.";
        rule succeeds.

A behavior rule:
    if platter is carried by Alison and the player is in the Dining Room and almost all of the supporters are not concerned:
        let the next victim be the red velvet chair;
        move the platter to the next victim;
        say "Whistling a jaunty tune, Alison brings in a hot platter and deposits it on [the next victim], then returns to the kitchen.";
        rule succeeds.

A behavior rule:
    if the tile is carried by Alison and the player is in the Dining Room and almost all of the supporters are not concerned:
        let the next victim be a random contented chair;
        move the tile to the next victim;
        say "Humming to herself, Alison brings in [the tile] and sets it down smack on [the next victim], then goes back out.";
        rule succeeds.

A behavior rule:
    if kettle is carried by Alison and the player is in the Dining Room and at least three fixed in place supporters are contented:
        let the next victim be the fussy table;

```
        move the kettle to the next victim;
        say "Just then the kettle boils in the kitchen.
    Whistling chirpily, Alison brings it in and sets it down
    smack on [the next victim], then goes back out.";
        rule succeeds.
```

And just so that the player knows where he stands at the end of each turn:

```
Section 6 - General Assessment

An every turn rule:
    if some of the things are concerned, say "You sense
    some resentment from [the list of concerned things]."

An every turn rule:
    if the player is carrying more than one scalding thing:
        let the dropped item be a random scalding thing
    carried by the player;
        say "You can't hang onto so many burning hot
    things, and lose your grip on [the dropped item].[line
    break]";
        silently try dropping the dropped item.

An every turn rule:
    if almost all of the supporters are concerned,
        say "'I hope everything is going well,' bellows
    Alison from the other room."
```

Finally, we need some rules to decide when the player is, in fact, finished.

```
Section 7 - Scoring

An every turn rule:
    now score is 5 minus the number of concerned
    things;
```

```
    if the location is concerned, decrement the score;
    if all of the supporters are concerned and the location
is concerned, end the story;
    if none of the supporters are concerned and the
location is not concerned, end the story finally.



The maximum score is 5.



When play begins, now score is 4.



Test me with "i / take off left shoe / get the glass / put
the glass on the platter / get the tile / drop the tile / get
the kettle / put the kettle on tile / sit on the blue
armchair".
```

---

112 **Example** **A Day For Fresh Sushi** *WI*

A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is performed.

The following is an almost-completely-faithful rewrite of Emily Short's "A Day for Fresh Sushi", which was originally written using the (very different) Inform 6 programming language. The -- let us be honest and call it a gimmick -- of this game is the evil fish, who has some unpleasant remark to offer on pretty much every action. But the effect would wear off fast if he repeated himself, so these comments need to be single-use only.

Inform 7's repeated action syntax makes it much tidier to write the same scenario, so:

```
"A Day For Fresh Sushi" by Emily Short.



Use scoring.
```

The story headline is "Your basic surreal gay fish romance".

The Studio is a room. "[if visited]Decorated with Britney's signature flair. It was her innate sense of style that first made you forgive her that ludicrous name. And here it is displayed to the fullest: deep-hued drapes on the walls, the windows flung open with their stunning view of old Vienna, the faint smell of coffee that clings to everything. Her easel stands over by the windows, where the light is brightest.[otherwise]This is Britney's studio. You haven't been around here for a while, because of how busy you've been with work, and she's made a few changes -- the aquarium in the corner, for instance. But it still brings back a certain emotional sweetness from the days when you had just met for the first time... when you used to spend hours on the sofa...

You shake your head. No time for fantasy. Must feed fish.[end if]"

Instead of smelling the Studio:
    say "The evil fish notices you sniffing the air. 'Vanilla Raspberry Roast,' it remarks. 'You really miss her, don't you.'

You glance over, startled, but the fish's mouth is open in a piscine equivalent of a laugh. You stifle the urge to skewer the thing..."

Instead of jumping:
    say "'Er,' says the fish. 'Does that, like, EVER help??'"

Instead of going nowhere:
    say "You can't leave until you've fed the fish. Otherwise, he'll complain, and you will never hear the end of it."

The cabinet is an openable closed container in the Studio. It is fixed in place. "A huge cabinet, in the guise of an armoire, stands between the windows." The description is "Large, and with a bit of an Art Nouveau theme going on in the shape of the doors." Understand "armoire" as the cabinet.

Instead of looking under the cabinet for the first time:
    say "'Dustbunnies,' predicts the fish, with telling accuracy. It executes what for all the world looks like a fishy shudder. 'Lemme tell you, one time I accidentally flopped outta the tank, and I was TWO HOURS on the floor with those things STARING ME IN THE NOSE. It was frightening.'"

After opening the cabinet for the first time:
    say "'There ya go,' says the fish. 'The girl is getting WARMER.'"

After closing the cabinet for the first time:
    if the fish food is not found, say "'Ooh, what do you think, Bob? I think we're going to have to dock the girl a few points. HAVE ANOTHER LOOK, sweetcakes, there's a doll.'"

The cabinet contains some paints and some cloths. The description of the paints is "A bunch of tubes of oil paint, most of them in some state of grunginess, some with the tops twisted partway off."

After taking the paints for the first time:

say "'Boy,' says the fish, apparently to himself, 'I sure hope that's some food she's finding for me in there. You know, the yummy food in the ORANGE CAN.'"

After examining the paints for the first time:
say "'Tons of useful stuff in there,' hollers in the fish, in a syncopated burble."

The description of the cloths is "Various colors of drapery that Britney uses to set up backgrounds and clothe her models. She does a lot of portraiture, so this comes in handy. It's all a big messy wad at the moment. Organized is not her middle name." Understand "drapery" or "cloth" as the cloths. The indefinite article of the cloths is "a heap of". [see 3.17]

Instead of searching or looking under the cloths for the first time:
now the player is carrying the fish food;
now the fish food is found;
say "Poking around the cloths reveals -- ha HA! -- a vehemently orange can of fish food."

Instead of showing the cloths to the fish:
say "'What are you, some kind of sadist? I don't want to see a bunch of cloths! What kind of f'ing good, 'scuse my French, is that supposed to do me? I don't even wear pants for God's sake!'

He really looks upset. You start wondering whether apoplexy is an ailment common to fish."

After examining cloths for the first time:
say "'Whatcha looking at? I can't see through the doors, you know.'"

There is a can of fish food. Understand "canister" as the can. The description is "A vehemently orange canister of fish food." The fish food can be found or hidden. The fish food is hidden.

Instead of giving the can to the fish:
    say "'I don't want the whole can, GeniusChyk. Just feed me and we'll ALL be happy, 'kay?"

Instead of showing the can to the fish:
    say "'That's the ticket, sweetie! Bring it on.'"

Instead of opening the can:
    say "'Oh, for--!' The evil fish breaks out in exasperation and hives. 'Screw the screwing around with the screwtop. SHE never has to do that.'

'Well, SHE is not here,' you reply. 'What do you suggest?'

'>FEED FISH<' says the fish promptly, making fishy faces and pointing at you with his fin. 'Simplicity. Try it.'"

Instead of inserting the can into something:
    say "'HelLLLOOO,' screams the fish. 'Whatever happened to FEEDING MEEE?'"

The easel is a supporter in the Studio. It is scenery. On the easel is a painting. Understand "portrait" or "image" as the painting.

The description of the painting is "Only partway finished, but you can tell what it is: Britney's mother. You only met the old woman once, before she faded out of existence in a little hospice in Salzburg.

In the picture, her hands are grasping tightly at a small grey bottle, the pills to which she became addicted in her old age, and strange, gargoyle-like forms clutch at her arms and whisper in her ears.

But the disturbing thing, the truly awful thing, is the small figure of Britney herself, down in the corner, unmistakable: she is walking away. Her back turned.

You thought she'd finally talked this out, but evidently not. Still feels guilty for leaving. You only barely stop yourself from tracing, with your finger, those tiny slumped shoulders..."

Instead of taking the painting, say "No, you'd better leave it. It'd freak her out if you moved it."

Before examining the painting for the first time:
say "A ferocious banging from the aquarium attracts your attention as you go to look at the painting. 'Hey!' screams the fish. 'She doesn't like strangers looking at her paintings before they're DOONNNE!'

'Shut up, you,' you reply casually. 'I'm not a stranger.' But the fish puts you off a little bit, and your heart is already in your mouth before you see the painting itself...".

Instead of examining the painting more than once:
    say "Once is really enough. It's pretty much embedded in your consciousness now."

After doing something to the painting:
    say "'So what's it of?' asks the fish, as you turn away. 'She never asks if I want to see them, you know?'

'Her mother,' you respond without thinking.

'Yeah? Man. I never knew my mother. Eggs, that's the way to go.'"

The window is scenery in the Studio. The window can be openable. The window can be open. It is openable and closed. Understand "windows" as the window. The description of the window is "[if open]Through the windows you get a lovely view of the street outside. At the moment, the glass is thrown open, and a light breeze is blowing through.[otherwise]Through the windows, you get a lovely view of the street outside -- the little fountain on the corner, the slightly dilapidated but nonetheless magnificent Jugendstil architecture of the facing building. The glass itself is shut, however.[end if]"

After opening the window for the first time:
    say "'Thank god some air,' says the fish. 'Man, it was getting hard to breathe in here.' Two beats pass. 'Oh wait.'"

The table is scenery in the Studio. On the table is a vase. The vase is an open container. It is not openable.

The description of the table is "A monstrosity of poor taste and bad design: made of some heavy, French-empire sort of wood, with a single pillar for a central leg, carved in the image of Poseidon surrounded by nymphs. It's all scaley, and whenever you sit down, the trident has a tendency to stab you in the knee. But Britney assures you it's worth a fortune." The description of the vase is "A huge vase -- what you saw once described in a Regency romance as an epergne, maybe -- something so big that it would block someone sitting at the table from seeing anyone else also sitting at the table. But it does function nicely as a receptacle for hugeass bouquets of flowers."

Instead of looking under the table for the first time:
    say "'You're not going to find anything down there,' whines the fish. 'I mean, c'mon. It's the fricking floor. Please tell me you can see that. I can see that. I'm a myopic fish in a tank ten feet away and I can tell you there is nothing there but floor.'"

After examining the table:
    say "'That there is MY PA,' says the fish, pointing at the scaley triton figure with one fin."

Instead of inserting something which is not the bouquet into the vase:
say "'Okay, so, what were you, raised in a barn? Normal folks like to use that for flowers. Or so I've observed.'"

After inserting the bouquet into the vase for the first time:
    say "You settle the flowers into the vase and arrange them so that they look sprightly.

'Oooh,' says the fish. 'No one ever changes the plant life in HERE. It's the same seaw--'

'Cut me a break and cork it,' you reply tartly."

The player is carrying a telegram, a bouquet, and a lingerie bag. The player is wearing a chef hat.

The description of the telegram is "A telegram, apparently. And dated three days ago. [fixed letter spacing]TRIUMPH OURS STOP BACK SOON STOP BE SURE TO FEED FISH STOP[variable letter spacing]". [For printing options see 4.13.] Understand "yellow paper" as the telegram.

After examining the telegram for the first time:
    say "'So,' blubs the evil fish. 'How about it? Little food over here?'"

After examining the telegram:
    choose a random row in the Table of Insulting Fish Comments;
    say "[comment entry][paragraph break]".

Table of Insulting Fish Comments
comment
"'Yeah, yeah,' says the fish. 'You having some trouble with the message, there? Confused? Something I could clear up for you?'"
"'Ookay, genius kid has some troubles in the reading comprehension department.' The fish taps his head meaningfully against the side of the tank. 'I'm so hungry I could eat my way out, you get my meaning?'"
"'I'll translate for you,' screams the fish in toothy fury. 'It says GIVE FOOD TO FISH!! How much more HELP do you NEED???"

The description of the chef hat is "A big white chef hat of the kind worn by chefs. In this case, you. Just goes to

show what a hurry you were in on the way out of the restaurant." Understand "big" or "white" or "chefs" or "chef's" as the chef hat. [Inform knows that this is clothing because the player starts out wearing it, so there's no need to say so separately.]

The aquarium is a transparent open container in the Studio. It is not openable. "In one corner of the room, a large aquarium bubbles in menacing fashion." The description of the aquarium is "A very roomy aquarium, large enough to hold quite a variety of colorful sealife -- if any yet survived." Understand "tank" as the aquarium.

The aquarium contains some gravel and some seaweed. Understand "little rocks" as the gravel. Understand "weed" as the seaweed. The description of the gravel is "A lot of very small grey rocks." The description of the seaweed is "Fake plastic seaweed of the kind generally bought in stores for exactly this purpose."

The examine containers rule does nothing when examining the aquarium.

After examining the gravel for the first time:
    say "The fish notices your gaze; makes a pathetic mime of trying to find little flakes of remaining food amongst the gravel."

After examining the seaweed for the first time:
    say "'Nice, hunh?' blubs the fish, taking a stabbing bite out of one just by way of demonstration. 'Look so good I could eat it.'"

The aquarium contains an animal called an evil fish. The description of the fish is "Even if you had had no prior

experience with him, you would be able to see at a glance that this is an evil fish. From his sharkish nose to his razor fins, every inch of his compact body exudes hatred and danger."

Instead of taking the evil fish:
    say "The fish swims adroitly out of range of your bare hand. 'Hey,' he says, and the bubbles of his breath brush against your fingers. 'Count yourself lucky I don't bite you right now, you stinking mammal.'"

Instead of attacking the evil fish:
    say "Oh, it's tempting. But it would get you in a world of hurt later on."

Instead of kissing the evil fish:
    say "You're saving all your lovin for someone a lot cuddlier."

After examining the evil fish for the first time:
    say "The fish glares at you, as though to underline this point."

After examining the evil fish for the second time:
    say "'If you're looking for signs of malnutrition,' says the fish, 'LOOK NO FURTHER!!' And it sucks in its gills until you can see its ribcage."

An every turn rule:
    choose a random row in the Table of Fish Banter;
    say "[comment entry][paragraph break]".

Table of Fish Banter

assessment
"Hey, nice SKIN TONE,' shouts the evil fish. His words reach you in a spitting gurgle of aquarium water. 'You gone over to a pure eggplant diet these days?'"
"The evil fish is floating belly up! ...oh, curse. He was toying with you. As soon as he sees you looking, he goes back to swimming around."
"The evil fish darts to the bottom of the tank and moves the gravel around with his nose."
"The evil fish is swimming around the tank in lazy circles."
"The evil fish begins to butt his pointy nose against the glass walls of the tank."

The description of the bouquet is "Okay, so it's silly and sentimental and no doubt a waste of money, of which there is never really enough, but: you miss her. You've missed her since ten seconds after she stepped aboard the shuttle to Luna Prime, and when you saw these -- her favorites, pure golden tulips like springtime -- you had to have them." Understand "flowers" or "tulip" or "tulips" as the bouquet.

After examining the bouquet for the first time:
    say "'Oh, you shouldn't have,' says the fish. 'For me??'
    You just respond with a livid glare."

    Instead of smelling the bouquet for the first time:
        say "'Mmm-mm,' says the fish. 'Damn, I sure wish I had olfactory abilities. Hey, if I did, I might be even better at noticing the presence or absence of FOOD.'"

    The description of the lingerie bag is "You grant yourself the satisfaction of a little peek inside. You went with a pale, silky ivory this time -- it has that kind of sophisticated innocence, and it goes well with the purple of your skin. A small smirk of anticipation crosses your lips."

After examining the lingerie bag for the first time:
say "'What's in THERE?' asks the fish. 'Didja bring me take-out? I don't mind Chinese. They eat a lot of carp, but what do I care? I'm not a carp. Live and let live is what I s--'

'It's NOT take-out.' You stare the fish down and for once he actually backstrokes a stroke or two. 'It's PRIVATE.'"

After examining the lingerie bag for the second time:
say "'If it's not take-out, I don't see the relevance!' shouts the fish. 'Food is what you want in this situation. Food for MEEEE.'"

Understand the command "feed" as something new.

Understand "feed [something]" as feeding.

Feeding is an action applying to one visible thing.

Check feeding:
if the noun is not the evil fish, say "That doesn't make much sense." instead;
if the player is not carrying the fish food, say "You need the fish food first." instead.

Carry out feeding:
increment the score;
say "Triumphantly, you dump the remaining contents of the canister of fish food into the tank. It floats on the surface like scum, but the fish for once stops jawing and starts eating. Like a normal fish. Blub, blub.[paragraph break]";
say "[bold type] *** TWO HOURS LATER ***[roman

type][paragraph break]'So,' Britney says, tucking a strand of hair behind your ear, 'where shall we go for dinner? Since I made the big bucks on this trip, it's my treat. Anywhere you like.'[paragraph break]'I've had a hankering all day,' you admit, as the two of you turn from the shuttle platform and head toward the bank of taxis. 'I could really go for some sashimi right now.'";
    end the story finally.

Before feeding the fish food:
    try feeding the evil fish instead.

When play begins:
    say "You're on the run. You've got a million errands to do -- your apartment to get cleaned up, the fish to feed, lingerie to buy, Britney's shuttle to meet-- [paragraph break]The fish. You almost forgot. And it's in the studio, halfway across town from anywhere else you have to do. Oh well, you'll just zip over, take care of it, and hop back on the El. This'll be over in no time.[paragraph break]Don't you just hate days where you wake up the wrong color?[paragraph break]".

The maximum score is 1.

Test me with "x fish / g / kiss fish / x aquarium / x gravel / x seaweed / i / x telegram / x bouquet / smell bouquet / x lingerie / g / x hat / x window / open window / x painting / g / x cabinet / open cabinet / x cloths / search cloths / open food / feed fish".

ExampleBribery

A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

If we want to rewrite the functionality of a command that usually ends

with a "block..." rule, we will have to begin by turning the blocking off.

```
"Bribery"

The block giving rule is not listed in the check giving it to
rules.
```

As it happens, correct behavior is built into the GIVE command once "block giving" is turned off, so we do not have to write a replacement report or carry-out rule; the object will be transferred to the possession of the caterpillar. But we do want to adjust the action just a little so that our gift cheers up the recipient:

```
Carry out giving (this is the gratitude for gifts rule):
improve the mood of the second noun.

Mood is a kind of value. The moods are hostile,
suspicious, indifferent, friendly, and adoring. An animal
has a mood. An animal is usually indifferent.

To improve the mood of (character - an animal):
    if the mood of character is less than friendly, now the
mood of the character is the mood after the mood of the
character.
```

Now whenever we give something to an animal, the animal will be pleased about the present. Of course, we might also want to add a check rule to giving, to see whether the offering is something the recipient really wants:

```
Check giving (this is the polite refusal of unwanted
objects rule):
    unless the noun interests the second noun:
        say "[The second noun] disdainfully refuses [the
noun]." instead.
```

```
To decide whether (item - a thing) interests (character -
a person):
    if the character has the item, no;
    if the item is edible, yes;
    no.
```

```
Instead of showing something to someone:
try giving the noun to the second noun.
```

There is already a perfectly workable report rule that will describe what happens when we give something to someone, but let's say we want to report on the recipient's changed mood, too:

```
After giving something to someone:
    say "You give [the noun] to [the second noun], who
appears mollified and is now merely [mood of the
second noun]."
```

And the rest is all scenario:

```
The Leafy Branch is a room. "You stand on smooth bark
dappled by sunlight. The scent-trail runs forward to
home.
```

```
The branch continues forward and backward from here,
and a stem extends forward-up."
```

```
Instead of going south in Leafy Branch, say "You must
not back down! The scent trail leads onward!"
```

```
The Very Hungry Caterpillar is a hostile animal in the
Leafy Branch. "[The Caterpillar] looks [mood]." Instead
of examining the Caterpillar, say "[The caterpillar]
appears [mood]."
```

The player carries an edible thing called a peanut crumb. The carrying capacity of the player is 1. After taking something, say "You lift [the noun], though it is nearly your own size."

Instead of going north in the presence of a hostile caterpillar:
    say "[The Caterpillar] moves to block your exit, glaring down at you with all the bristles on its skin extended to full size."

Instead of going north in the presence of a suspicious caterpillar:
    say "[The Caterpillar] moves to block your exit, though it might allow you past if you offered further tribute."

The Leaf Face is above the branch. "The smooth and shiny surface of the leaf extends forward from here, but you have lost the scent-trail. This is not the way home." The pear fragment is an edible thing in Leaf Face. The dead aphid is a thing in Leaf Face.

The Twig is north of Leafy Branch. "The scent-trail is weak but not entirely gone, and you pursue it faithfully..."

After going to the Twig:
    say "The scent-trail is weak but not entirely gone, and you pursue it faithfully...";
    end the story finally.

Understand "forward-up" as up. Understand "forward" as north. Understand "backward" as south. Understand

"backward-down" as down.

Test me with "forward / give crumb to caterpillar / forward / forward-up / get aphid / get fragment / down / give aphid to caterpillar / drop aphid / forward-up / get fragment / down / give fragment / forward".

ExampleBarter Barter

Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

By default, if we make no modifications, telling one player to give something to another will fail, even if persuasion succeeds. This is because the default behavior of the GIVE command is interrupted by the "block giving rule" -- since in many cases we do not want people to exchange objects freely.

However, suppose that we do want characters to be able to exchange articles freely: we allow persuasion to succeed and turn off the "block giving rule".

"Barter Barter"

The block giving rule is not listed in the check giving it to rules.

A persuasion rule for asking people to try giving: persuasion succeeds.

The Trading Post is a room.

```
Meriwether Lewis is a man in the Trading Post. He
carries a fluffy handmade quilt and a bag of beans. The
beans are edible.

William Clark is a man in the Trading Post. He carries
leather slippers, a journal, and a loaf of bread. The
bread is edible. The slippers are wearable.

Instead of examining someone:
    say "[The noun] is carrying [the list of things carried
by the noun]."
```

And now we might want to implement a way to keep track of whether the recipient character wants what's being offered:

```
Check someone trying giving something to someone
(this is the sneering refusal rule):
    if the second noun dislikes the noun, stop the action.

Unsuccessful attempt by someone trying doing
something:
    if the reason the action failed is the sneering refusal
rule, say "'Would you care for [the noun]?' [the person
asked] asks solicitously of [the second noun].

But [the second noun] refuses [the noun] disdainfully.";
    otherwise say "[The person asked] just appears
bewildered by this improbable instruction."

Distaste relates one person to various things. The verb
to dislike means the distaste relation.

Clark dislikes the beans. Lewis dislikes the bread.
```

Since we've defined this as a relation, we could change what the characters like and dislike during the course of the game, freely; for instance, characters might grow hungry and suddenly like all the edible articles.

```
Test me with "x lewis / x clark / clark, give the slippers to
lewis / clark, give the bread to lewis".
```

---

130 **Example** **Lanista 1** *WI*

Very simple randomized combat in which characters hit one another for a randomized amount of damage.

```
"Lanista, Part One"


The Arena is a room. "Sand, blood, iron. These festivals
are normally held on hot days, but the sun has gone
behind a cloud and fat drops of rain now and then
spatter the arena floor." The gladiator is a man in the
Arena. "A bare-chested Scythian gladiator faces you,
wielding a trident."
```

We start by recording, for each person, a maximum number of points of damage the person can sustain when starting from health, and the current number of points remaining. In the tradition of role-playing games, these are referred to as hit points.

```
A person has a number called maximum hit points. A
person has a number called current hit points.


The maximum hit points of the player is 35. The
maximum hit points of the gladiator is 25.
```

```
The current hit points of the player is 35. The current hit
points of the gladiator is 25.
```

Now our rule for the actual attack. We want first to calculate how much damage the player's attack does, inflict that damage, and remove the enemy if he's dead; then, if he doesn't die, the enemy counter-attacks, also for a randomized amount of damage, and if this kills the player, the game ends in defeat.

```
Instead of attacking someone:
    let the damage be a random number between 2 and
10;
    say "You attack [the noun], causing [damage] points
of damage!";
    decrease the current hit points of the noun by the
damage;
    if the current hit points of the noun is less than 0:
        say "[line break][The noun] expires, and is
immediately carried away by the Arena slaves!";
        now the noun is nowhere;
        end the story finally;
        stop the action;
    let the enemy damage be a random number between
2 and 10;
    say "[line break][The noun] attacks you, causing
[enemy damage] points of damage!";
    decrease the current hit points of the player by the
enemy damage;
    if the current hit points of the player is less than 0:
        say "[line break]You expire!";
        end the story.
```

This last bit is a refinement to help the player keep track of how the contest is going:

```
When play begins:
    now the left hand status line is "You: [current hit
points of player]";
    now the right hand status line is "Gladiator: [current
hit points of gladiator]".
```

```
Test me with "hit gladiator / g / g / g".
```

A DIAGNOSE command which allows the player to check on the health of someone.

```
"Red Cross"

Diagnosing is an action applying to one visible thing.
Understand "diagnose [something]" as diagnosing.

Check diagnosing:
    if the noun is not a person, say "Only people can
have diagnoses." instead.

Carry out diagnosing:
    say "You're not a medical expert."

Instead of diagnosing the player:
    say "You feel fine!"

The Clinic is a room. Fred is a man in the Clinic. The
player carries a tongue-depressor.

Test me with "diagnose me / diagnose fred / diagnose
tongue-depressor".
```


A system in which every character has a body, which is left behind when the person dies;

attempts to do something to the body are redirected to the person while the person is alive.

Suppose we want to let the player kill characters, leaving behind corpses.

```
"Puff of Orange Smoke"


Paraguay is a room. Bolivia is north of Paraguay. Lydia
is a woman in Paraguay. "Lydia is, as usual, here." The
description of Lydia is "Long, long legs and a sarcastic
attitude." Instead of touching Lydia: say "'Watch it,
sailor,' she snaps."


A body is a kind of thing. A body is a part of every
person. Instead of touching a body: say "[The noun] is
grotesquely inert."


The description of Lydia's body is "Long, long legs and
no attitude at all." The initial appearance of Lydia's body
is "Lydia's corpse is sprawled at your feet."
```

Using our "part of every person..." line, we've conveniently assigned one body per person. Since we're going to separate people from their bodies when the bodies die, though, we also want a more permanent relation that will help us keep track of which bodies used to belong to which people:

```
Spirit-possession relates one person to one body. The
verb to be owner of means the spirit-possession
relation.


When play begins:
    repeat with victim running through people:
        let the corpse be a random body which is part of
the victim;
        now the victim is owner of the corpse.
```

When Lydia is alive, we want >TOUCH LYDIA'S BODY to mean the same thing as >TOUCH LYDIA, so we use the setting action variables rules as a convenient point at which to reassign the action:

```
Setting action variables when the noun is a body which
is part of a person (called owner):
    now the noun is the owner.
```

```
Setting action variables when the second noun is a body
which is part of a person (called owner):
    now the second noun is the owner.
```

This doesn't change Inform's idea about what action is being performed; just about the object it's being performed on. The rest of the action will now proceed as if the player had typed >TOUCH LYDIA.

Along similar lines, once Lydia is dead, we want >MOVE LYDIA to mean >MOVE LYDIA'S BODY if the body is in view:

```
Setting action variables when the noun is a dead person
and the noun is owner of a visible body (called the
mortal remains):
    now the noun is the mortal remains.
```

The trick is, though, that >MOVE LYDIA will only be understood if there is something called Lydia that the player can see and refer to, even after she's dead. There are various ways to do this, but the least painful here will be to make the deceased Lydia permanently visible, by putting her in an always-accessible backdrop. The backdrop itself will never be mentioned in the game, and we should make its name something that the player is unlikely to type casually; we don't want the player to interact with it directly. So:

```
The worldview is a privately-named backdrop. It is
everywhere. The spirit-world is a privately-named
transparent closed unopenable container. It is part of
the worldview.
```

```
Definition: a person is dead if he is in the spirit-world.
```

It's also possible that the player will type something like >X LYDIA when Lydia's corpse is not in view, so we should have an appropriate answer to that as well:

```
Before doing something to a dead person:
    say "[The noun] is dead; or had you blocked that
out?" instead.
```

Because the before rules happen after the setting action variables rules, this will only ever happen if the corpse is not visible.

Now we define the attack itself, which should discard the body, move the spirit to its eternal resting place, and describe the event to the player:

```
Instead of attacking someone:
    let the corpse be a random body which is part of the
noun;
    move the corpse to the location;
    move the noun to the spirit-world;
    say "With a single blow, you rid the world of [the
noun]."
```

And finally a trick borrowed from the chapter on understanding, so that we can refer to "Lydia's body" while Lydia is alive, but "Lydia's corpse" only after Lydia has died:

```
Understand "corpse" as a body when the item described
is not part of a person.
```

```
Test me with "x body / x lydia's body / touch lydia's body
/ x corpse / kill lydia / look / x lydia's body / x lydia's
corpse / x corpse / x lydia / touch lydia / lydia, hello / n /
x lydia / touch lydia / lydia, hello".
```


Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an ATTACK IT WITH action is created to replace regular

attacking. Also folds a new DIAGNOSE command into the system.

Back in the chapter on randomization, we explored a way to create a randomized combat system. That system didn't allow for multiple weapons, though. Here we explore how to create an ATTACK IT WITH action that will let the player choose between weapons with different maximum powers.

We're also going to rewrite that original "instead of attacking:" rule into an attacking it with action that can be performed equally by the player or by any of the player's enemies.

```
"Lanista, Part Two"


The Arena is a room. "Sand, blood, iron. These festivals
are normally held on hot days, but the sun has gone
behind a cloud and fat drops of rain now and then
spatter the arena floor." The gladiator is a man in the
Arena. "A bare-chested Scythian gladiator faces you,
wielding [a list of weapons carried by the gladiator]."


Section 1 - Hit Points


A person has a number called maximum hit points. A
person has a number called current hit points.


The maximum hit points of the player is 35. The
maximum hit points of the gladiator is 25.
```

In our simpler version of this example we set the current hit points by hand, but in a game with many characters this would get dull and repetitive, so here we'll use a "when play begins" to set all current hit point values automatically to maximum:

```
When play begins:
```

repeat with victim running through people:
now the current hit points of the victim is the maximum hit points of the victim.

Definition: a person is dead if his current hit points are less than 0.

Section 2 - Diagnosis

Diagnosing is an action applying to one visible thing. Understand "diagnose [something]" as diagnosing.

Check diagnosing:
if the noun is not a person, say "Only people can have diagnoses." instead.

Carry out diagnosing:
say "[if the noun is the player]You have[otherwise][The noun] has[end if] [current hit points of the noun] out of a possible [maximum hit points of the noun] hit points remaining."

Section 3 - Weapons

A weapon is a kind of thing. A weapon has a number called the maximum damage. The maximum damage of a weapon is usually 4.

The gladiator carries a weapon called a trident. The maximum damage of the trident is 5. The gladiator carries a weapon called a net. The maximum damage of the net is 1.

```
The player carries a weapon called a mace. The
maximum damage of the mace is 3.
```

Section 4 - Attacking it with

In our new system, we want to specify what is being used for an attack. This means that we need to create a new "attacking it with" action, and also that we should disable the existing "attacking..." command.

Here's why: If we leave the default attack command in place, Inform will continue to accept commands like >ATTACK GLADIATOR, but reply foolishly with the default "Violence is not the answer..." response.

A somewhat better approach would be to change the reply of >ATTACK GLADIATOR to say something like "You must specify a weapon to attack with." But this is still less than ideal, because it means that the player has to then rewrite his entire command. If, on the other hand, we take out "ATTACK GLADIATOR" entirely, the game will always prompt "What do you want to attack the gladiator with?" -- which teaches the player the correct command structure for this particular game, and avoids pretending to understand any command that is not meaningful within this game.

This is a little bit of work because ATTACK has a lot of synonyms in the default library, but if we look through the actions index we can find them all:

```
Understand the commands "attack" and "punch" and
"destroy" and "kill" and "murder" and "hit" and "thump"
and "break" and "smash" and "torture" and "wreck" as
something new.
```

Now we make our new command:

```
Attacking it with is an action applying to one visible thing
and one carried thing. Understand "attack [someone]
with [something preferably held]" as attacking it with.
```

Note that we've specified "one carried thing", because we want the player to pick up a weapon to use if necessary. And now we assign all the old attack vocabulary to apply to the new command:

```
Understand the commands "punch" and "destroy" and
"kill" and "murder" and "hit" and "thump" and "break"
and "smash" and "torture" and "wreck" as "attack".
```

This may seem counter-intuitive, but order of source code matters here: we first get rid of the old, default vocabulary, then define our new action, then make the vocabulary apply to that new action. Inform will now understand >HIT GLADIATOR WITH TRIDENT, >BREAK GLADIATOR WITH TRIDENT, and so on.

Our new action is also a perfect place to use an action variable: we're going to need to choose an amount of damage done and refer to that several times in our action rules. So let's set that up first:

```
The attacking it with action has a number called the
damage inflicted.


Setting action variables for attacking something with
something:
    if the second noun is a weapon:
        let the maximum attack be the maximum damage
of the second noun;
        now the damage inflicted is a random number
between 1 and the maximum attack.


Check an actor attacking something with something
(this is the can't attack with something that isn't a
weapon rule):
    if the second noun is not a weapon:
        if the actor is the player, say "[The second noun]
does not qualify as a weapon.";
        stop the action.


Check an actor attacking something with something
(this is the can't attack a non-person rule):
    if the noun is not a person:
        if the actor is the player, say "[The noun] has no life
to lose.";
        stop the action.
```

```
Carry out an actor attacking something with something
(this is the standard attacking it with a weapon rule):
    decrease the current hit points of the noun by the
damage inflicted;
    if the noun is dead:
        now the noun is nowhere.
```

Though our checks and carry-out rules are similar regardless of who is acting, we're going to want actions to be described differently for different actors, so we'll use separate "report attacking" and "report someone attacking" rules. We'll also make some special cases for when the character has died as a result of the attack:

```
Report attacking a dead person with something (this is
the death-report priority rule):
    say "You attack with [the second noun], killing [the
noun]!" instead.
```

```
Report attacking someone with something (this is the
normal attacking report rule):
    say "You attack [the noun] with [the second noun],
causing [damage inflicted] point[s] of damage!" instead.
```

```
Report someone attacking the player with something
when the player is dead (this is the player's-death
priority rule):
    say "[The actor] attacks you with [the second noun],
finishing you off!";
    end the story;
    stop the action
```

```
Report someone attacking the player with something
(this is the standard report someone attacking the player
with rule):
    say "[The actor] attacks you with [the second noun],
causing [damage inflicted] point[s] of damage!" instead.
```

```
Report someone attacking something with something
(this is the standard report attacking it with rule):
    say "[The actor] attacks [the noun] with [the second
noun], causing [damage inflicted] point[s] of damage!"
instead.


When play begins:
    now the left hand status line is "You: [current hit
points of player]";
    now the right hand status line is "Gladiator: [current
hit points of gladiator]".


Every turn (this is the gladiator-attack rule):
    if the gladiator is not dead, try the gladiator attacking
the player with a random weapon which is carried by the
gladiator.


Test me with "hit gladiator with mace / kill gladiator /
drop mace / attack gladiator / attack gladiator with mace
/ g / g".
```

Those devoted to role-playing will note that our form of randomization is still pretty naive: most RPG systems use multiple dice in order to create more interesting probability curves. For a system that simulates actual dice-rolling, see the full "Reliques of Tolti-Aph" game.

Example **Don Pedro's Revenge** *WI*

Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

Suppose our game features a detailed simulated combat between the player character and his opponent. He might have several weapons available, and several types of attack available; and at any given time he might be perched up in the rigging of his ship, standing on the open deck, or boxed in between some barrels. His options will vary depending on his position, and obviously it would detract from the pacing to make the player keep LOOKing in the middle of combat in order to remind himself where he is. Instead, we'll roll this information into the command prompt:

"Don Pedro's Revenge"

The Deck of the Helene Marie is a room. "The two crews are embattled all around you, but your attention is reserved for your particular enemy: Don Pedro."

Table of Random Prompts

position	prompt
boxed	"So securely boxed-in that you can really only parry or thrust, you try to "
trapped	"Trapped between your barrels, you decide to "
perched	"Perched up here with the advantage of height (but little mobility), you attempt to "
free	"Out on the open deck with no impediments, free to advance or retreat, you decide to "

When play begins: reset the prompt.

Every turn: reset the prompt.

To reset the prompt:
    sort the Table of Random Prompts in random order;
    repeat through the Table of Random Prompts:
        if the position entry is the placement of the player:
            now the command prompt is prompt entry;
            stop.

After reading a command: say conditional paragraph break.

A placement is a kind of value. The placements are boxed, perched, free. The player has a placement. The player is free.

Understand "retreat" or "parry" as retreating. Retreating is an action applying to nothing.

Check retreating:
    if the player is perched, say "You can't move backward or parry very successfully from this position." instead.

Carry out retreating:
    now the player is boxed;
    say "You protect yourself, but end up wedged in between two barrels."

Understand "thrust" or "advance" as advancing. Advancing is an action applying to nothing.

Check advancing:
    if the player is perched, say "You can't move forward from here, only slash." instead.

Carry out advancing:
    now the player is free;
    say "You push forward aggressively, making your way to the open deck."

Instead of jumping:
    now the player is perched;
    say "You leap and swing yourself boldly up into the

```
rigging, leaving your attackers beneath you."


Instead of jumping when the player is perched:
    now the player is free;
    say "You leap down from your position, into the
middle of the deck."


Test me with "advance / jump / advance / retreat / jump
/ retreat / retreat / advance".
```

Of course, this won't be much fun until we also provide the player with a few weapons, some more fighting maneuvers, and, most of all, a Don Pedro to defeat.

ExampleTechnological Terror

A ray gun which destroys objects, leaving their component parts behind.


```
"Technological Terror"


The Decomposition Ray Gun is a thing carried by the
player.
```

First we need to define our shooting action:

```
Shooting it with is an action applying to two things.


Check shooting something with something:
    if the player is not carrying the Ray Gun, say "You are
pathetically unarmed!" instead;
    if the second noun is not the Ray Gun, say "[The
second noun] does not fire." instead;
```

```
    if the noun is the Ray Gun, say "Nice trick if you can
do it!" instead;
    if the noun is the player, say "That would be
disastrous!" instead.
```

Next, some grammar to allow the player to use this action:

```
Understand "shoot [gun] at [something ungunlike]" as
shooting it with (with nouns reversed).

Definition: a thing is ungunlike if it is not the gun.

Understand "shoot [something ungunlike] with [gun]" as
shooting it with. Understand "shoot [something] with
[something]" as shooting it with.

Understand "shoot [something] at [something]" as
shooting it with (with nouns reversed). Understand "fire
[gun] at [something ungunlike]" as shooting it with (with
nouns reversed). Understand "fire at [something
ungunlike] with [gun]" as shooting it with. Understand
"fire at [something] with [something]" as shooting it with.
```

Strictly speaking, we only need these last grammar lines (with "understand shoot something...") in order to define an action that the player can take. Adding more grammar lines means that Inform will try to match the most specific ones first, which is useful when the player types something ambiguous and there is one choice that obviously fits this action better than the others. See the chapter on Understanding for a further discussion.

Here we get to use "now..." to give it its destructive effect:

```
Carry out shooting something with something:
    say "ZAP! [The noun] twinkles out of existence! [if
something is part of the noun][The list of things which
are part of the noun] clatter to the ground! [end if]
[paragraph break]";
    now every thing which is part of the noun is in the
```

location;
    now the noun is nowhere.

The Deathbot Assembly Line is a room. "Here is the heart of the whole operation, where your opponents are assembled fresh from scrap metal and bits of old car." The dangerous robot is a thing in the Assembly Line. "One dangerous robot looks ready to take you on!" A robotic head, a drill arm, a needle arm, a crushing leg and a kicking leg are parts of the dangerous robot.

Instead of examining something when something is part of the noun:
    say "[The noun] consists of [a list of things which are part of the noun]."

Test me with "x robot / shoot robot / fire at kicking leg / shoot gun at drill arm / look".

Example **The Gorge at George**

WI

If the player tries to TALK TO a character, suggest alternative modes of conversation.

"Gorge at George"

The Dusty Lot is a room. "A few miles up the road from the concert venue, but at least it's cheap to park here."

The motorcyclist is a man in the Dusty Lot. "A man clad in [a list of things worn by the motorcyclist] leans against his Harley and watches you without saying anything." The Harley is scenery in the Lot. The motorcyclist wears

a black leather jacket and shades. Understand "man" or "guy" as the motorcyclist.

Understand "talk to [someone]" as a mistake ("To start a conversation, try to ASK [the noun] ABOUT something or TELL [the noun] ABOUT something.").

Instead of asking the motorcyclist about something:
    say "He smirks cryptically."

Instead of telling the motorcyclist about something:
    say "This does not seem to interest him much."

Test me with "talk to motorcyclist / ask motorcyclist about himself / tell motorcyclist about me".

---

110 ExampleMimicry *WI*

People who must be greeted before conversation can begin.

Suppose we want to add a sense of some conversational flow, so that the player is forced to acknowledge the presence of people before beginning detailed conversations with them. We collect all speech actions into a single category:

"Mimicry"

Asking someone about something is speech. Telling someone about something is speech. Answering someone that something is speech. Asking someone for something is speech.

And then write a general rule.

```
Before speech in the presence of an ungreeted person:
try waving hands.
```

One complication is that "asking someone to try doing something", which describes commands such as FRED, GO SOUTH, cannot be made into a kind of action. This requires its own rule:

```
Before asking someone to try doing something in the
presence of an ungreeted person: try waving hands.
```

Now we define what greetings are going to look like:

```
Check waving hands:
    unless the player can see someone who is not the
player, say "You are alone." instead.

Carry out waving hands:
    say "You nod hello to [the list of ungreeted people
who can be seen by the player].";
    now every ungreeted person who can be seen by the
player is greeted.

The report waving hands rule is not listed in the report
waving hands rulebook.
```

Because of the way we've defined the command, this will now also work if the player waves.

```
A person can be greeted or ungreeted. A person is
usually ungreeted. The player is greeted.
```

And now the scenario:

The International Convention of Mimes is a room. Lester, Harold, Geoff, Kwame, and Peter are men in the Convention. Elouise is a woman in the Convention. The Invisible Box is an enterable container in the Convention. "You can detect, from the way people keep leaning on it, an invisible box in the middle of the room."

Lester carries a bowler hat.

Instead of speech in the presence of someone:
    describe poor reception.

Definition: a person is other if it is not the player.

At 9:01 AM:
    move Phineas to the location; say "A mime called Phineas appears from the non-existent bathroom."

Phineas is a man.

A persuasion rule:
    describe poor reception;
    persuasion fails.

To describe poor reception:
    if the player is in the Invisible box,
        say "Everyone convulses with silent laughter as you try to shout from within the invisible box.";
    otherwise
        say "You attempt to convey your meaning with gesture and interpretive dance, but [the list of visible other people] scorn[if the number of visible other people is 1]s[end if] your performance, refusing to respond."

Test me with "ask lester about work / lester, east / ask lester for bowler / lester, nice not talking to you / get in box / ask lester for hat / phineas, east".

Example Sybil 1 *WI*

Direct all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

Sometimes we do not particularly want to deal with all the variations on asking, telling, or answering someone something, but want to direct everything to a single conversational command:

"Consulting the Oracle"

The Grove is a room. In the Grove is a woman called the Sybil.

Instead of telling someone about something, try asking the noun about it. Instead of answering the noun that something, try asking the noun about it.

Instead of asking the Sybil about "persians", say "She nods gravely."

And similarly, a difference between GIVE and SHOW is sometimes overkill:

Instead of showing something to someone, try giving the noun to the second noun.

```
The player carries a coin. Instead of giving the coin to
the Sybil: move the coin to the Sybil; say "She accepts
with a smile."
```

It is also often the case that we want to accept more than one form of a term. For instance

```
Instead of asking the Sybil about "Darius/king", say "Her
smile unnerves you."
```

will match either "Darius" or "king". If necessary, we can go a step further and define our own token to match a variety of phrases, like this:

```
Understand "Athenians/Spartans/Greeks" or "hoplite
army/forces" as "[Greeks]". Instead of asking the Sybil
about "[Greeks]", say "She looks encouraging."
```

The token "[Greeks]" will match all of "Athenians", "Spartans", "Greeks", "hoplite army", or "hoplite forces". It will not match "hoplite" or "forces" alone; it is important to note that the / divides individual words which are understood equivalently, but does not define entire phrases as equivalent. More about how Inform understands specific phrases can be found in the chapter on Understanding.

```
Test me with "test one / test two".
```

```
Test one with "ask sybil about persians / tell sybil about
persians / sybil, persians / ask sybil about darius / ask
sybil about king".
```

```
Test two with "ask sybil about greeks / ask sybil about
athenians / ask sybil about hoplite army / ask sybil about
hoplite forces / give the coin to the sybil".
```

Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

Inform already understands YES, NO, and SORRY as commands in their own right, which can make things a little sticky when we want a character to ask a question of the player. The most important thing is not to cover some of the possible phrasings while ignoring others.

"Replies"

The Grove is a room. In the Grove is a woman called the Sybil.

Instead of asking the Sybil to try saying no: try saying no. Instead of asking the Sybil to try saying yes: try saying yes. Instead of asking the Sybil to try saying sorry: try saying sorry.

Instead of answering the Sybil that "yes", try saying yes.
Instead of answering the Sybil that "no", try saying no.
Instead of answering the Sybil that "sorry", try saying sorry.

Instead of saying yes in the presence of the Sybil:
    say "She looks interested."

Instead of saying no in the presence of the Sybil:
    say "She looks annoyed."

Instead of saying sorry in the presence of the Sybil:
    say "She looks bored."

The complexity arises from the fact that we want to handle both YES and SYBIL, YES. If we only had the latter, 'yes' would be treated as a text given to the Sybil, just as in the commands SAY YES TO SYBIL or ANSWER YES. But because we have defined it as a command (so that the player can use it independently), SYBIL, YES is understood as an order to the Sybil to do the YES action.

Fortunately, we can redirect everything, as here, so that the results wind up the same.

And if we want yet another variation not covered by the Inform standard:

```
Understand "tell [someone] [text]" as answering it that.
Understand "tell [someone] that [text]" as answering it
that.
```

But that is a matter for a later chapter.

```
Test me with "yes / sybil, yes / say yes to sybil / answer
yes / tell sybil yes / no / sybil, no / say no to sybil /
answer no / tell sybil no / sorry / sybil, sorry / say sorry
to sybil / answer sorry / tell sybil sorry".
```

ExampleProposal

Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

Suppose we want to ask the player a question where he might say yes or no in response. There are two possible forms of this: the modal question where the player must pick one to proceed, and the non-modal question where he might also type other verbs.

```
"Proposal"
```

```
The story genre is "A Worked Example about Yes/No
Questions".
```

Section 1 - Asking a Modal Yes/No Question

When play begins:
    say "Do you like Mr Spruce? ";
    if player consents, now Spruce is handsome;
    otherwise now Spruce is ugly;
    say paragraph break.

Section 2 - Mr Spruce's Non-Modal Question

Use full-length room descriptions.

The Conservatory is a room. "You are in a room full of plants."

Mr Spruce is a man in the Conservatory. Mr Spruce can be apprehensive or calm. Mr Spruce is calm. Mr Spruce can be handsome or ugly.

At 9:02 AM: say "Mr Spruce flings himself to his knees and implores you to become his lawfully wedded wife.";
    now Mr Spruce is apprehensive;
    Mr Spruce gives up in two minutes from now.

At the time when Mr Spruce gives up:
    say "Mr Spruce sighs heavily, seeing that you don't intend to reply. 'Never mind, my dear, I'll ask later. Perhaps I should have spoken to your Papa first... yes, a gently-bred female... no wonder...'";
    now Mr Spruce is calm;
    Mr Spruce departs in one minute from now.

At the time when Mr Spruce departs:
    if the player can see Mr Spruce, say "Mr Spruce takes his leave of you.";
    otherwise say "Mr Spruce pokes his head in to say that he is leaving.";
    end the story saying "Well, that is over..."

Instead of saying yes in the presence of an ugly apprehensive Mr Spruce:
    now Mr Spruce is calm;
    say "Remembering what your mother said to you about the stock exchange and Dear Papa, you close your eyes and accept Mr Spruce.";
    end the story saying "Alas for your maiden hopes."

Instead of saying yes in the presence of a handsome apprehensive Mr Spruce:
    now Mr Spruce is calm;
    say "You are silent with delight for a moment before you say yes, yes!";
    end the story saying "How Genevieve Stubbs will cry!"

Instead of saying no in the presence of an ugly apprehensive Mr Spruce:
    now Mr Spruce is calm;
    say "Gently you inform Mr Spruce that it is impossible. He seems less deflated than you had expected.";
    end the story saying "Odd, that..."

Instead of saying no in the presence of a handsome apprehensive Mr Spruce:
    now Mr Spruce is calm;
    say "You lower your eyes and refuse petulantly, hoping to stir him to a more ardent repetition of these same requests. But then -- alack! -- he says 'I see how it is!' in a strangled voice, and strides from the room!";
    end the story saying "A fatal error!"

And since the player might SAY YES TO SPRUCE, we had better reroute the relevant options:

```
Instead of answering Mr Spruce that "no", try saying no.

Instead of answering Mr Spruce that "yes", try saying
yes.

Instead of asking Mr Spruce to try saying yes, try saying
yes.

Instead of asking Mr Spruce to try saying no, try saying
no.

Instead of saying sorry, try saying no.

Instead of asking Mr Spruce to try saying sorry, try
saying no.

Instead of answering Mr Spruce that "sorry", try saying
no.

Test me with "z / z / z / yes".

Test more with "z / z / z / no".
```

ASKing someone about an object rather than about a topic.

By default, ASK SOMEONE ABOUT... applies only to a text token. We might want also to offer the player the option of asking characters about pieces of physical evidence. This example implements an ASK PERSON ABOUT THING command that is mostly synonymous with SHOW, with the added nuance that the player can ask about things that are not currently visible, as long as he has encountered them at some time in the past.

```
"Nameless"

The Black Chamber is a room. "Despite its menacing
name, it is quite an ordinary room, underlying the post
office above. Here letters are brought each day,
unsealed, transcribed, resealed, and sent again on their
way; their contents then analyzed and recorded."

The Nameless Advisor is a woman in the Black
Chamber. "A woman whose name has never been
disclosed to you sits at the window, writing numbers on
a sheet of paper." The Advisor carries a sheet of paper.
Understand "woman" as the nameless advisor.

The player carries a letter from the emperor. The
description of the letter is "Though its origin is obvious,
its meaning is secret: the letters are an inexplicable
jumble."
```

Now we create our new action, "interrogating it about". We write the grammar lines so that we can show any object in sight to someone, but also ask someone about any object that we have ever interacted with in the game, whether it is currently visible or not.

```
A thing can be known or unknown. The Nameless
Advisor is known.
```

```
Understand "ask [someone] about [any known thing]" as
interrogating it about. interrogating it about is an action
applying to two visible things.
```

Now we replace and redirect the showing action. This gets rid of the requirement in the default library that the player be holding anything he shows to another character:

```
Understand the commands "show" and "display" and
"present" as something new.
```

```
Understand "show [something] to [someone]" or "display
[something] to [someone]" or "present [something] to
[someone]" as interrogating it about (with nouns
reversed). Understand "show [someone] [something]"
as interrogating it about.
```

This bit keeps track of what the player has seen, for the purposes of "any known thing":

```
Before printing the name of something (called the
target): now the target is known.
```

Here we define what happens by default when we interrogate someone about something; we use the same response we get to asking someone about something that isn't otherwise interesting:

```
Carry out interrogating someone about something:
    say "There is no reply."
```

Now redirect all asking to a topic table, and all interrogating to an object table:

```
Instead of asking Nameless Advisor about a topic listed
in the Table of Nameless Advisor Topics:
    say "[reply entry][paragraph break]".


Instead of interrogating Nameless Advisor about an item
listed in the Table of Nameless Advisor Items:
    say "[reply entry][paragraph break]".


Table of Nameless Advisor Items

item                reply
"letter"            "'Ah, a letter enciphered,' she remarks[if the advisor can see the letter], glancing over the contents[otherwise], after you have offered a detailed description[end if]. 'A substitution cipher of some complexity, I believe.'"
"Advisor"           "She listens to your inquiries about her identity and parentage with a placid smile, but does not answer."


Table of Nameless Advisor Topics

topic               reply
"cipher"            "'I know many dozens of ciphers,' she replies, smiling in a disquieting way."
"substitution"      "'One letter is allowed to stand for another,' she explains, folding her hands together patiently. The backs of both hands are tattooed with silvery stars."
```

And just so that we can test what happens when asking someone about something out of sight:

```
The safe box is a container in the Chamber. It is fixed in
place. It is openable and closed.
```

...and something unknown:

```
The poisonous apple is a thing.
```

Test me with "test sight / test knowledge".

Test sight with "i / x letter / ask Nameless Advisor about cipher / show cipher to Nameless Advisor / ask Nameless Advisor about the letter / show the letter to Nameless Advisor / show Nameless Advisor the letter".

Test knowledge with "open safe box / put letter in safe box / close safe box / ask Nameless Advisor about the letter / show the letter to Nameless Advisor / ask Nameless Advisor about the apple".

---

281 ExampleFarewell

WI

People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.

We begin with the idea that each person comes with his own table of things to say:

"Farewell"

A person has a table name called conversation.

```
Instead of asking someone about something:
    let the source be the conversation of the noun;
    if topic understood is a topic listed in source:
        if there is a turn stamp entry:
            say "[The noun] has already told you that
[summary entry].";
        otherwise:
            now turn stamp entry is the turn count;
```

```
            say "[reply entry][paragraph break]";
        otherwise:
            say "[The noun] stares at you blankly."
```

For the sake of simplicity, we'll conflate asking and telling here, though it would certainly be possible to have a more complex implementation if we want the characters to be told things as well.

```
Instead of telling someone about something:
    try asking the noun about it.
```

Now we might want to add a recap command to review conversation that has already occurred.

```
Definition: a person is other if it is not the player.
```

```
Understand "recap" or "recall" or "review" as recalling
conversations.
```

```
Recalling conversations is an action applying to nothing.
```

Since we've been recording the turn count of each conversation bit, we can even present these in order by sorting the tables first.

```
Carry out recalling conversations:
    repeat with speaker running through other people:
        let source be the conversation of the speaker;
        sort source in turn stamp order;
        say "[The speaker] has so far told you: [line
break]";
        let index be 0;
        repeat through source:
            if there is a turn stamp entry:
                let index be 1;
                say " [summary entry][line break]";
        if index is 0, say " absolutely nothing[line break]";
        say conditional paragraph break.
```

Now it remains only to create a couple of characters and provide them

both with something to say:

> The Farewell Bend Cafe is a room. "Beautiful Farewell Bend, Idaho -- or is it Oregon? An almost-abandoned truckstop, in any case, on one of those interminable American east-west highways."

> Tina is a woman in the Farewell Bend Cafe. The conversation of Tina is the Table of Tina's Chatter. "Tina the waitress is slowly pouring coffee from the pot with a black neck into the pot with an orange neck."

> George is a man in the Farewell Bend Cafe. The conversation of George is the Table of George's Chatter. "There is also a large man at table five. The tattoo on his arm says George. For the moment we will assume that it is his own name and not someone else's."

Table of Tina's Chatter

[illegible]	[illegible]
[illegible]	"[illegible] is a vulture"
[illegible]	"[illegible] something doesn't have a tail.'"
[illegible]	"[illegible] explains gently, as to a child."

Table of George's Chatter

[illegible]	[illegible]
[illegible]	"[illegible] looking up."
[illegible]	"[illegible]"
[illegible]	"[illegible] replies instantly."

A word of warning: this system does assume that every person in the game has a conversation table defined. If that were not the case, we

would have to be a bit more careful.

As always, we can override specific words, too:

```
Instead of asking Tina about "advertisement" for the first
time:
    say "Tina looks embarrassed. 'Of course! I almost
forgot.' She hands you a brochure.";
    move the brochure to the player.


The encyclopedia sales brochure is a thing. The
description is "A glossy flyer indicating that you can
receive a free Volume A-Aalto of the New Idahoan
Encyclopedia Set if you send back the business reply
card, and then have the option of purchasing the
remaining volumes at a very very reasonable price."


Test me with "recap / ask tina about aasvogel / recap /
ask george about baccaceous / ask tina about
absorptiometer / recap / ask tina about advertisement /
read brochure".
```

ExampleSweeney

A conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

```
"Sweeney"


A subject is a kind of thing. Some subjects are defined
by the Table of Conversation Subjects.


Table of Conversation Subjects
```

subject
Table of Pie Queries
table of Job Queries

Understand "job" as employment. Understand "meat" or "food" as pies.

Table of Job Queries

"'Say, are you hiring?' you ask, as casually as you can manage. [The interlocutor] looks you over dubiously. 'I might be hiring someone, but I can't say it would necessarily be you.'"

"'What happened to that boy that was working here?' She shrugs. 'Young men these days are so unstable. He left-- who knows where he's gone? I haven't seen hair or fingernail of him for weeks.'"

with 3 blank rows.

Table of Pie Queries

"'What pies do you have today, Mrs Lovett?' you ask. She starts, then smiles. 'Meat pies, of course.'"

with 3 blank rows.

Table of Pie Flavor Queries

"'What kind of meat goes into these pies, Mrs Lovett?' you ask pressingly. She looks shifty. 'Whatever the butcher brings this week,' she says. 'With the price of meat what it is, when you get it, you have to be glad of what you can get. If you get it.'"

To copy (first table - a table name) to (second table - a table name):
    repeat through first table:
        let copied quip be "blank";
        if there is a quip entry, now the copied quip is the quip entry;
        let copied discussion be "blank";
        if there is a discussion entry, now the copied discussion is the discussion entry;
        let copied subtopics be second table;
        if there is a subtopics entry, now the copied subtopics are the subtopics entry;
        choose a blank row in the second table;
        if copied quip is not "blank", now quip entry is copied quip;
        if copied discussion is not "blank", now discussion entry is copied discussion;
        if copied subtopics is not second table, now subtopics entry is copied subtopics.

Current conversation table is a table name that varies.
Current conversation table is Table of Job Queries.

Interlocutor is a person that varies.

Understand "ask [someone] about [any subject]" as asking it about the subject.

Asking it about the subject is an action applying to two visible things.

Carry out asking it about the subject:
    say "You can't think of anything to say."

Instead of asking someone about the subject a subject listed in the Table of Conversation Subjects:

```
    now interlocutor is noun;
    now current conversation table is the conversation of
the second noun;
    if the number of filled rows in the current conversation
table is 1:
        repeat through current conversation table:
            now label entry is 1;
        now number understood is 1;
        try selecting 1 instead;
    if the number of filled rows in the current conversation
table is 0:
        say "You can think of nothing further to say on that
topic.";
        stop the action;
    otherwise:
        let index be 0;
        let total be the number of filled rows in the current
conversation table;
        say "Do you mean ";
        repeat through current conversation table:
            now index is index + 1;
            now label entry is index;
            say "([index]) [quip entry]";
            if index is total, say "?";
            if index is total - 1, say ", or ";
            if index is less than total - 1, say ", ".



Understand "[number]" as selecting.



Selecting is an action applying to one number.



Carry out selecting:
    say "No such option is available."



Instead of selecting a label listed in the current
conversation table:
    say "[discussion entry][paragraph break]";
    if there is a subtopics entry:
        copy subtopics entry to current conversation table;
    choose row with label of number understood in the
current conversation table;
```

```
blank out the whole row.


Mrs Lovett's Meat Pies is a room. Mrs Lovett is a
woman in Meat Pies.


Test me with "ask lovett about pies / ask lovett about
employment / 1 / 2 / ask lovett about pies".
```

ExampleCheese-makers

Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

As we have seen, there are a number of different ways of controlling conversation in interactive fiction, and the best choice of way will depend quite a lot on what kind of work we're writing.

One common model is to replace Inform's default ASK and TELL commands with a TALK TO command. This gives the player less control than he would otherwise have: instead of asking a character about any topic under the sun, he's restricted to seeing (or not seeing) a single sequence of text that the author has written in advance. On the other hand, such a system is harder for the player to break (since he can never ask about a topic that the author hasn't implemented), and easier for the author to tie into plot developments. If we give TALK TO different output at each scene, we get conversation that is always tied to the current state of the plot.

This is a design approach that works best in a game with a large number of short, focused scenes. For other kinds of conversation system design, compare the other examples listed in the Recipe Book.

```
"The Cheese-makers" by Phrynichus.


Chapter 1 - Replacing old talk commands and making a
new one
```

Here, using some techniques that will be discussed in the chapter on Understanding, we get rid of Inform's default handling of ASK and TELL, and create our own TALK TO action instead:

```
Understand the commands "ask" and "tell" and "say"
and "answer" as something new.

Understand "ask [text]" or "tell [text]" or "answer [text]"
or "say [text]" as a mistake ("[talk to instead]").

Instead of asking someone to try doing something:
    say "[talk to instead][paragraph break]".

Instead of answering someone that something:
    say "[talk to instead][paragraph break]".

To say talk to instead:
    say "(To communicate in [story title], TALK TO a
character.) "

Understand "talk to [someone]" as talking to.
Understand "talk to [something]" as talking to. Talking to
is an action applying to one visible thing.
```

Chapter 2 - Specific scenes and talking

Now, suppose we have a situation -- say, a stage play -- in which it is appropriate to talk to different characters at different times. During the prologue of the play, no one else is on-stage, and the player is to address the audience directly:

Section 1 - Prologue

When play begins:
    now right hand status line is "416 BC";
    now left hand status line is "[location]".

Prologue is a scene. Prologue begins when play begins.

The Theater of Dionysus is a room.

The audience is a person in the Theater. "The usual audience looks on: the priests and judges in the front row, and then Athenians, metics, and foreigners." The audience can be prepared or unprepared. The description is "Have you ever seen such a company of perjurers, pathics, and thieves?" Understand "priest" or "priests" or "priest of dionysus" or "judge" or "judges" or "athenians" or "metics" or "foreigners" as the audience.

Instead of talking to the player when the Prologue is happening:
    say "There will be plenty of occasion for muttered asides later in the play, but for now you must prepare the audience for things to come."

Instead of talking to the audience when the Prologue is happening:
    say "Drawing breath, you turn to the audience, and offer them a genial, witty, colorful, and of course crude synopsis of what they are about to see; describing all the characters in unmistakable terms and not omitting the most important of them all, your august self.";
    now the audience is prepared.

Instead of talking to the audience when the Prologue has happened:
    say "You may only direct monologues to the audience when the other actors are off-stage. Otherwise, their characters might have to notice."

Prologue ends when the audience is prepared.

But there might follow a scene in which the player shouldn't talk at all:

Section 2 - Parodos

Parodos is a scene. Parodos begins when Prologue ends.

When Parodos begins:
    move the chorus to the theater.

Instead of talking to someone during Parodos:
    say "Sssh: this moment belongs to the chorus. They've worked so hard on it, after all."

Parodos ends when the time since Parodos began is 4 minutes.

The chorus is a person. The description is "They are dressed in exaggerated rural costume and feminine masks, as they are meant to represent a company of female cheese-makers from the Spartan-occupied deme of Dekeleia." Understand "cheesewives" or "cheese-makers" or "chorus-leader" as the chorus.

```
Every turn during Parodos:
    repeat through Table of Choral Events:
        say "[output entry][paragraph break]";
        blank out the whole row;
        make no decision.
```

```
Table of Choral Events
output
"The chorus now begins its entry, accompanying with
anapestic song its march up the eisodos."
"The chorus draws nearer, stomping and clomping and
swinging their baskets of cheese."
"You stand aside as the chorus fills the orchestra and
dances to and fro."
"The tune of the aulos-player grows more and more
frenzied and then breaks off."
```

This last rule is a refinement borrowing from the Activities chapter, which gives characters different appearances in room descriptions depending on when we happen to look; because of the action of the play, we want to show the chorus and audience doing different things during different scenes.

```
Rule for writing a paragraph about the chorus during
Parodos:
    say "The chorus are dancing and singing their way[if
the time since Parodos began is less than 3 minutes] up
the long walkways onto the stage[otherwise] into
position in the orchestra[end if]. [The audience] appear
to be pricing their costumes to the nearest obol: woe to
the producer who cheats them of their due share of
spectacle."
```

And now a scene in which the player can talk several times to a character (Heracles) but has no useful dialogue with the chorus, the audience, or himself. The prohibition from talking to the audience after the Prologue is already written, but we'll supply some appropriate responses for talking to the player or the chorus during this scene:

```
Section 3 - Episode
```

Episode is a scene. Episode begins when Parodos ends.

When Episode begins:
    move Heracles to the theater;
    say "The chorus falls silent, which is the cue: Heracles bursts out of the scene building."

Heracles is a man. The description is "Hard to mistake in his lion skin and boots, and carrying a formidable club." Heracles wears a lion skin and boots. He carries a formidable club. Heracles can be placid or annoyed. Heracles is placid. Heracles can be satisfied, intrigued, or unsatisfied. Heracles is unsatisfied.

Instead of talking to the chorus during Episode:
    say "Your improvised flirtation with the chorus raises no response but a crude gesture from the chorus-leader, who seems to be modeling the role on Iambe."

Instead of talking to the player during Episode:
    if Heracles is annoyed:
        say "You mutter to yourself about men with more appetite than brain. The actor playing Heracles ignores you, but it's good odds he's scowling under his mask. He hates it when anyone but himself ad-libs for attention.";
    otherwise:
        now Heracles is annoyed;
        say "'By the dog, he'll eat me if he gets a chance,' you mutter aside. [paragraph break]'What's that you say, my ignoble friend?' demands Heracles, hefting his club. He's not entirely joking: you've left the script just now."

Instead of talking to Heracles when Heracles is unsatisfied during Episode:
    say "'Dear Heracles, friendly Heracles,' you begin, cringing out of the way as he responds with one of his

affectionate ox-killing punches to the shoulder. [paragraph break]But Heracles falls still, and looks almost thoughtful, as tell him you know how he may rout the Spartans, woo all twenty-four lactic ladies, and tame his savage gut with a bathtubful of porridge. [paragraph break]'Speak on, little man,' he says.";
    now Heracles is intrigued.

Instead of talking to Heracles when Heracles is intrigued during Episode:
    say "It takes several exchanges for him to wrap his one-inch brain around your ten-inch plan; but in the end he embraces the scheme, the women, and your humble self.";
    now Heracles is satisfied.

Every turn when not talking to someone during Episode:
    repeat through Table of Episodic Events:
        say "[output entry][paragraph break]";
        blank out the whole row;
        make no decision.

Table of Episodic Events
output
"With a fart and a roar, Heracles asks the world at large, and you in particular, where his dinner might be."
"In epic diction, Heracles invites the dairy-mistresses, whey-matrons, and concubines of curd to supply him a supper from their ample baskets."
"Heracles and the chorus banter about the proclivities of cheese-wives. The chorus suggest that Heracles, as a son of Zeus, must know something about the appetites of which they speak."
"Heracles boasts that a man like himself can perform any feat, but only when his belly is full. Coyly, the matrons prance and dance, skip and gambol out of his grasp, singing mockingly about heads of garlic and loaves of sesame-crusted bread."
"The song of the feta fanciers now turns to pots of honey and new-made wine, borrowing verses from last year's Lenaia winner, 'The Bees'. With a jolt, you realize that you've missed your cue and the chorus are filling in for you."
"Playing for time, the chorus-leader elaborates a whole

banquet: rabbit stew, shanks of lamb, spitted quails, eels from lake Copais. Heracles looks as near swooning as any girl fresh from Brauron."
"The chorus-leader extends the list of delicacies to include ox-brains, ham-hocks, barley, mullet, carrots, pigeons, lentils, radishes, peas, and apples both wine-dark and golden. The audience shifts on the benches. An expression of gloom settles over the Priest of Dionysus in the front row."
"Inspired by Euripides['] own Muse, the chorus-leader invents a mock-Alcaean hymn on the merits of chervil. This is clearly his swan-song: if you don't speak at last, the play will come to a halt."
"Silence descends."

Rule for writing a paragraph about Heracles during Episode:
    say "[Heracles] stands at the center of the orchestra, with members of [the chorus] ranged on either side. [paragraph break][The audience] appear to be reserving their judgement, though they show signs of restiveness at the usual jokes: must there be a Heracles in [italic type]every[roman type] play?"

Episode ends successfully when Heracles is satisfied.

When Episode ends successfully:
    say "That, of course, is your cue: you're to come back on as Pan thirty verses from now, and it takes time to put on the hooves and the woolly-legged trousers.";
    end the story saying "You exit".

Episode ends disastrously when the number of filled rows in the Table of Episodic Events is 0.

When Episode ends disastrously:
    end the story saying "The production has crashed to a halt".

Test me with "ask audience about me / tell audience about me / audience, hello / audience, jump / talk to me / talk to audience / g / talk to chorus / look / x heracles / talk to me / talk to audience / z / look / talk to heracles / g".

ExampleComplimentary Peanuts *WI*

A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

The "reading a command" activity is not the only point at which we can interact with snippets, as it happens; it is merely the most useful. "The player's command" can be consulted at other points, however, as in this example of your somewhat deaf (or distracted, or simply cussed) Aunt:

```
"Complimentary Peanuts"

Instead of asking Aunt Martha to try doing something:
    repeat through Table of Aunt Martha's Commentary:
        if player's command includes topic entry:
            say "[commentary entry][paragraph break]";
            rule succeeds;
    say "'Hmmf,' says Aunt Martha."
```

The topic understood is also a snippet, so that whenever one has been generated, we can treat it in the same way as "the player's command":

```
Asking someone about something is speech.
Telling someone about something is speech.
Answering someone that something is speech.
Asking someone for something is speech.

Instead of speech when the noun is Aunt Martha:
    repeat through Table of Aunt Martha's commentary:
```

```
        if the topic understood includes topic entry:
            say "[commentary entry][paragraph break]";
            rule succeeds;
    say "'Hmmf,' says Aunt Martha."
```

This is superior to checking "the player's command" because we do not want ASK MARTHA ABOUT FRENCH FRIES to trigger the "Martha" keyword, only the "french fries" keywords.

```
The Empyrean Shuttle Bay is a room. "From here you
have an excellent view of the colony world, which
looks... well, it looks discouragingly orange. But
terraforming is in progress."

Aunt Martha is a woman in the Empyrean Shuttle Bay. A
gleaming shuttle and a stack of rations are in the Shuttle
Bay. The shuttle is a vehicle. "Your shuttle awaits."

Table of Aunt Martha's Commentary

topic                commentary
"shuttle"            "Shuttles! I hate shuttles,' Aunt Martha grumbles. 'Give me an
                     airplane! AIRPLANE.'"
"airplane"           "Those were the days,' Aunt Martha agrees, plainly reliving the
                     days when she wore a blue-and-white uniform and passed out
                     packets of salted pretzels."
"rations"            "Do you think there are any peanuts in there?' she asks in a
                     wistful tone."

Test me with "martha, get in the shuttle / martha, for
pity's sake, do you see an airplane around here? /
martha, pass me the rations".
```

This means that Martha will respond to keywords regardless of the setting in which they occur. For instance:

```
>martha, get in the shuttle
"Shuttles! I hate shuttles," Aunt Martha grumbles. "Give
```

me an airplane! AIRPLANE."

>martha, for pity's sake, do you see an airplane around here?
"Those were the days," Aunt Martha agrees, plainly reliving the days when she wore a blue-and-white uniform and passed out packets of salted peanuts.

>martha, pass me the rations
"Do you think there are any peanuts in there?" she asks in a wistful tone.

This is not the stuff of which Loebner-winning chatbots are made, admittedly, but it is occasionally a useful alternative to stricter modes of command-parsing.

---

405 **Example** **Being Peter** *WI*

A set of rules determining the attitude a character will take when asked about certain topics.

Let's say that we're implementing a particularly irrational and volatile character. Some of the time she remains composed; some of the time she reacts with unexpected vehemence for reasons only partly related to what was said.

Moreover, her responses are divided between successful and failing outcomes, where success indicates that she's not too upset and failure means that she is distraught; we use this to determine how the rest of the room reacts.

"Being Peter"

The Drawing Room is a room. "The company is assembled here for champagne. Most of it, anyway: Mary is on the phone to her babysitter, Roger is keeping her anxious company, and Carol doesn't drink. But

```
everyone else."

Maggie is a woman in the Drawing Room.

The player wears a top hat.

Quizzing it about is an action applying to one thing and
one visible thing. Understand "ask [someone] about
[any thing]" as quizzing it about.

Instead of quizzing Maggie about something:
    follow the attitude rules;
    say "Everyone waits to see what the reaction will be:
[outcome of the rulebook].";
    if rule succeeded, say "There is general relief.";
    otherwise say "Everyone is pointedly silent."

The attitude rules are a rulebook. The attitude rules
have outcomes she stays calm (no outcome - default),
she gets angry (failure), she has a stroke (failure), she is
only mildly annoyed (success), and she is elated
(success).
```

Here we want Inform to consult every appropriate attitude rule until it gets to some answer; if an attitude rule does not provide a result, the default 'no outcome' will mean that we go on to the next rule, and so on.

```
A subject is a kind of thing. income, love life, and
children are subjects.

An attitude rule for quizzing Maggie about love life:
    she gets angry.
```

```
An attitude rule:
    if the player wears the top hat, she gets angry.
```

Now, as we saw, the 'no outcome' result will never be returned and printed as Maggie's reaction, precisely because it is "no outcome". Therefore, we provide a final attitude rule which will give her a default response to all statements:

```
The last attitude rule:
    she is only mildly annoyed.


Test me with "ask maggie about love / ask maggie
about income / take off hat / ask maggie about income".
```

There are plenty of contexts where we might want named outcomes for clarity but not want to print the results literally afterward.

Example **Ferragamo Again**

WI

Using the same phrase to produce different results with different characters.

Here we use phrases that match individual items where possible, and the general kind otherwise:

```
"Ferragamo Again"


The Break Room is a room. Vanessa, Tina, and Lisa are
women in the Break Room. Mark and Holman are men
in the Break Room.


Understand the commands "ask" and "tell" and "answer"
as something new.
```

Understand "talk about [any subject]" as talking about. Talking about is an action applying to one visible thing.

Understand "talk about [text]" as talking randomly about. Talking randomly about is an action applying to one topic. Carry out talking randomly about: say "Mostly you're interested in [the list of subjects]."

Carry out talking about something:
    now the previous subject is the noun.

Report talking about something:
    say "You chat for a while about [the noun]."

A subject is a kind of thing. Assyrian vowel sounds, designer handbags, and instant run-off voting are subjects. Understand "linguistics" and "mute" and "stop" as sounds. Understand "prada" and "tods" and "coach" and "carmen marc valvo" as designer handbags. Understand "reform" and "election" and "election fraud" and "two-party system" and "Diebold" as instant run-off voting.

To say (annoyed-person - a person) gestures in irritation:
    say "[The annoyed-person] sighs heavily. [run paragraph on]"

To say (annoyed-person - Vanessa) gestures in irritation:
    say "[The annoyed-person] takes off her glasses and polishes them on her sleeve. [run paragraph on]".

To say (annoyed-person - Holman) gestures in irritation:

```
    say "Holman bobs his head. [run paragraph on]"

The previous subject is a subject that varies.

Instead of talking about something for more than one
turn:
    if the noun is the previous subject, say "[a random
visible person who is not the player gestures in
irritation]Maybe you should let this one go.[line break]
[paragraph break]";
    otherwise continue the action.

Test me with "talk about chocolate / talk about vowel
sounds / g / talk about handbags / talk about prada / talk
about tods".
```

Example **Murder on the Orient Express** *WI*

A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but there is physical evidence which may change their minds...

The following example creates two new relations, and two new verbs, in order to set up a tangled web of intrigue.

```
"Murder on the Orient Express"

The Dining Car is a room. Lord Peter is a man in the
Dining Car. Sherlock Holmes is a man in the Dining Car.
Miss Marple is a woman in the Dining Car. Adam
Dalgliesh is a man in the Dining Car.
```

Suspecting relates various people to one person.

The verb to suspect means the suspecting relation.

Dalgliesh suspects Holmes. Holmes suspects Lord Peter. Lord Peter suspects Holmes. Miss Marple suspects the player.

Exculpating relates one thing to various people.

The verb to exculpate means the exculpating relation.

The silver bullet exculpates the player. The pipe ash exculpates Holmes. The poison pen letter exculpates Lord Peter. The poison pen letter exculpates Miss Marple. [Poor Dalgliesh. I guess he did it.]

The pipe ash, the letter and the silver bullet are carried.

Given this, we can then set up elaborate rules:

Instead of showing something to a person who suspects the player:
    say "'You would say that,' remarks [the second noun] darkly.".

Instead of showing something which exculpates the player to someone:
    say "'How striking!' says [the second noun]. 'Almost I begin to distrust myself.'".

```
Test me with "show the letter to miss marple / show the
silver bullet to holmes".
```

And so on: "if Dalgliesh suspects someone who is exculpated by something carried by the player...", for instance, makes a fitting final example for this chapter. The description

```
someone who is exculpated by something carried by the
player
```

expresses a complicated idea in very few words, and in such a way that a passer-by looking at the source text would immediately see what was meant.

The moral is that relations allow sophisticated patterns of behaviour to be created in a way that reads back naturally as English.

Example **The Problem of Edith** *WI*

A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before.

Suppose that we have a core set of issues we want to be able to bring up with all the characters, and we want characters to draw intelligent connections between different conversation topics. We will need some model of how things relate to one another, so:

```
"The Problem of Edith"
```

```
Suggestion relates things to each other. The verb to
suggest means the suggestion relation.
```

```
A subject is a kind of thing. The current subject is a
thing that varies. greeting is a subject.
```

```
Understand "ask [someone] about [any subject]" as
asking it about the subject. Understand "tell [someone]
about [any subject]" as asking it about the subject.


Asking it about the subject is an action applying to one
thing and one visible thing.


Carry out asking it about the subject:
    say "'Hmm, [the second noun],' says [the noun]. ";
    relate the current subject with the second noun;
    now the current subject is the second noun.
```

And if we wanted to offer the player some hints about angles he could pursue:

```
Instead of thinking:
    say "You contemplate [a list of things suggested by
the current subject]."
```

For that matter, we could use the same system to have characters make sense of any physical evidence the character shows them:

```
Instead of showing something which suggests the
current subject to someone:
    say "[The second noun] nods impatiently."


Instead of showing something to someone:
    let the next subject be the next step via the
suggestion relation from the noun to the current subject;
    if the next subject is a subject:
        try asking the second noun about the subject the
next subject;
    otherwise:
        say "[The second noun] shrugs."
```

```
When play begins:
    now the left hand status line is "Discussing: [current
subject]";
    now the right hand status line is " ".
```

```
Broughton Hall is a room. Lady Uckfield is a woman in
Broughton Hall. "Lady Uckfield sits at her desk, looking
wholly composed."
```

```
The nasty letter is a thing carried by the player. The
nasty letter suggests infidelity and penmanship. The
ten-pound note is carried by the player. It suggests
money.
```

```
Infidelity is a subject. Infidelity suggests marriage and
divorce. Marriage suggests love. Marriage, love, and
divorce are subjects.
```

```
Penmanship is a subject. Penmanship suggests
education. Education is a subject. Class status and
money are subjects. Class status suggests education.
Money suggests class status and marriage.
```

```
The current subject is divorce.
```

Now we can define what gets said when the subject is changed, regardless of whether the segue was introduced in speech or by a shown object. Since rows are blanked after use, the speaker will never repeat herself; if we provide more than one line about the same pair of topics, the first one will be used, then the second, and so on, until the table runs out:

```
To relate (initial - a subject) with (next - a subject):
    repeat through Table of Remarks:
        if the initial is starting entry and the next is the final
entry:
```

```
        say "[comment entry][paragraph break]";
        blank out the whole row;
        rule succeeds;
    say paragraph break.
```

```
Table of Remarks

[comment entry]
[paragraph break]
"As it seems to me, all the love is on one side,' she says crisply. 'And that rarely works.'"
"Stop making that plea: it won't work.'"
"Frankly, I rather think there would have been cause enough for divorce without the perversely plentiful evidence of unfaithfulness.'"
"I may mean that the divorce will be expensive, I know it,' she says. 'But I can think of no happier investment.'"
"If you wish me to understand that it was a marriage for money, you could have spared your energy. That was patent from the outset.'"
"I'm sorry, but I don't see how having married for money excuses a subsequent infidelity.'"
```

If we had more than one character in the scenario, we could provide multiple tables, but this will do to demonstrate the idea.

Of course, we can override specific instances, if we want the character always to say the same thing regardless of how we came to this point:

```
Instead of asking Lady Uckfield about the subject penmanship:
    now the current subject is penmanship;
    say "She sighs. 'So few people write really beautifully these days.'"
```

```
Test me with "think / ask lady about infidelity / show nasty letter to lady / show note to lady / think / ask lady about divorce / ask lady about love / ask lady about marriage / ask lady about divorce / ask lady about love / ask lady about penmanship".
```

We would have to be careful about this system, since we have applied a various-to-various relation to every single object in the game. In practice it would probably be wisest to restrict it a bit, with judicious definitions of

kind and so on.

Example: Questionable Revolutions

An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

"Questionable Revolutions"

Interrogative is a kind of value. The interrogatives are who, what, when, where, how, and why.

Current question is an interrogative that varies.

After asking someone about something: respond to the question. After answering someone that something: respond to the question.

After telling someone about something: say "You're here to ask questions."

Country is a kind of value. The countries are Czechoslovakia, Georgia, Sweden, Italy, Spain.

Table of Information

[illegible]
[illegible]lution in Czechoslovakia, in which popular protests led to the resignation of the communist president Gustav Husak, and the election of Vaclav Havel in his place."

2000 revolution" which President Eduard Shevardnadze was interrupted by protesters in the middle of his speech, and forced to flee."

1936 revolution" which socialist movement during the Spanish civil war."

After reading a command:
    if the player's command includes "[interrogative]",
now the current question is the interrogative understood.

To respond to the question:
    repeat through the Table of Information:
        if the topic understood includes topic entry:
            if the current question is what or the current question is who, say definition entry appropriately;
            if the current question is when, say date entry appropriately;
            if the current question is where, say place entry appropriately;
            rule succeeds;
    say "[The noun] shrugs."

Understand "ask [someone] [text]" as asking it about.

Comprehension is a kind of value. The comprehensions are vague, erroneous, and correct.

Table of Understanding

[illegible]	[illegible] comprehension
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

```
When play begins:
    say "Here you are in the first class cabin, but no
matter how fancy the seats are, you can still get bored
circling over Zurich for three hours on end. To kill time,
you and the other passengers are playing a trivia game,
and the final topic is your specialty: revolutions."


First Class Cabin is a room. Dr Tweedy is a man in First
Class. Ms Finch and Ms Clarion are women in First
Class.


To say (year - a number) appropriately:
    choose row with character of the noun in the Table of
Understanding;
    if years entry is correct:
        say "'[year],' replies [the noun] promptly.";
        increment the quiz score of the noun;
    if years entry is erroneous:
        let guess be a random number between 1900 and
2005;
        say "'[guess]?' guesses [the noun], with an air of
diffidence[if guess is the year]. Which is right, as it
happens[end if].";
        if guess is the year, increment the quiz score of the
noun;
    if years entry is vague:
        let offset be a random number between -5 and 5;
        let year be year + offset;
        say "'I think [year]. About then. Close, anyway,'
replies [the noun][if the offset is 0], getting it right[end
if].";
        if offset is 0, increment the quiz score of the noun.


To say (spot - a country) appropriately:
    choose row with character of the noun in the Table of
Understanding;
    if geography entry is correct:
        say "'[spot],' replies [the noun] promptly.";
        increment the quiz score of the noun;
    if geography entry is erroneous:
        let guess be a random country;
        say "'Er... [guess]?' says [the noun][if guess is the
spot]. Which is of course correct[end if].";
        if guess is the spot, increment the quiz score of the
```

```
noun;
    if geography entry is vague, say "'Europe,' replies [the
noun] with confidence."



To say (explanation - some text) appropriately:
    choose row with character of the noun in the Table of
Understanding;
    if general comprehension entry is correct:
        say "'[explanation]'[paragraph break]";
        increment the quiz score of the noun;
    otherwise:
        choose a random row in the Table of Information;
        say "'[definition entry]'[paragraph break]";
        if the definition entry is explanation:
            say "[A random other person who is not the
noun] looks surprised that this came out right.";
            increment the quiz score of the noun.



A person has a number called quiz score. The quiz
score of Tweedy is 48. The quiz score of Finch is 2. The
quiz score of Clarion is 4.



Definition: a person is other if it is not the player.



When play begins:
    now left hand status line is "T: [quiz score of Tweedy]
F: [quiz score of Finch] C: [quiz score of Clarion]";
    now right hand status line is "[time of day]".



Test me with "dr tweedy, where was the velvet
revolution located / ms finch, when was the rose
revolution / ms finch, what was the rose revolution / ms
clarion, when was the spanish revolution / g / g / ms
finch, when was the spanish revolution".
```

We have so far seen several ways to write conversational characters in Inform, and we will see more before the end of the manual. This naturally raises the question, which should we use? To which the answer is: it depends on the sort of game we're writing, and what we want our

characters to do. The more rich and complex the system, the more likely that it will require a lot of content; if we add question types as well as keywords, for instance, we instantly multiply the number of responses we have to write by five or six. It is not worth doing this unless there is some corresponding advantage within the game.

ExampleThe Queen of Sheba

Allowing the player to use question words, and using that information to modify the response given by the other character.

Suppose we want the player to ask questions of slightly more complexity - we might want to build in a system that understood "who", "what", "where", and "when", for instance. We could use a topic table for this, too:

"The Queen of Sheba"

Interrogative is a kind of value. The interrogatives are who, what, when, where, how, and why.

Current question is an interrogative that varies.

After asking someone about something: respond to the question. After answering someone that something: respond to the question.

After telling someone about something: say "You're here to ask questions and test Solomon's wisdom, not to give him a sample of your own."

Table of Wise Answers

topic	question type	reply
"clouds/cloud"	what	"'Clouds are made by the paths of birds,' Solomon replies. 'The air beaten by their wings becomes agitated, as when a river is stirred and the mud churns up.'"
"weather/rain/clouds"	where	"'Water is contained in a great silk bag which holds in the heavens,' replies Solomon."
"hunger/food/eating"	where	"'Hungry, are you getting hungry?' he says, and rings a bell to summon servants."
"hunger/food/eating"	why	"'Men were made to need food in order that they must farm and cook and dine together,' Solomon replies. 'Otherwise, they might live apart, each sufficient in himself. But no man can feed himself alone all through his life.'"
"solomon/he/himself"	who	"'As you see,' he says, holding out his arms to each side."
"solomon/he/himself"	what	"'An ordinary man,' he answers."

One of the nice things about this system is that it only resets the "current question" when we get a new question word. For instance, this test will produce different replies to the question about Solomon himself, because the second time he is still in the mode of answering "what" questions:

```
Test me with "ask solomon about himself / ask solomon
what rain is / ask solomon about himself".
```

If Solomon is to live up to his reputation at all, his wisdom table will have to be quite a bit longer - though one also would want to be careful, because forcing the game to cycle through a really immense table could be quite time-consuming. In fact, for the sake of this example, let's reward the player for managing to stay within the (narrow) range of Solomon's knowledge:

```
The Hall of Almug Tree Pillars is a room. "The pillars of
the room are made of almug tree, the ceiling made of
silk and the floor of glass." Solomon is a man in the Hall
of Almug Tree Pillars. Solomon has a number called
wisdom. The wisdom of Solomon is 0.
```

```
Every turn:
    if the wisdom of Solomon is 3:
        say "Truly, Solomon has answered all your
questions, and his wisdom is even as great as you had
heard!";
        end the story saying "Your heart beats strangely
```

```
fast".


When play begins, say "'Oh, you've arrived,' says
Solomon."
```

In a real game we'd need to be a great deal subtler. All the same, if we have a character of quite limited resources to present to the player, it's a good idea to give the player some incentive to stay on topic, ask questions the character can answer, and generally interact within the parameters we're prepared for.

Now, this last bit requires some trickery from later chapters, particularly those on Understanding and Activities, to pull the question words out of the player's command:

```
After reading a command:
    if the player's command includes "[interrogative]",
now the current question is the interrogative understood.


To respond to the question:
    repeat through the Table of Wise Answers:
        if the topic understood includes topic entry:
            if the current question is the question type entry:
                say "[reply entry][paragraph break]";
                increment the wisdom of Solomon;
                rule succeeds;
    say "Solomon looks blank, appalled by a question for
which he was not prepared.";
    end the story saying "You have befuddled Solomon!"


Understand "ask [someone] [text]" as asking it about.
```

And now we have a game that will accept (though not always respond very sensibly to) questions of almost any form we might put to another character: ASK SOLOMON WHAT RAIN IS will be answered, but then again, it won't be distinguished from, say, ASK SOLOMON WHETHER THIS PERSISTENT RAIN IS A DIVINE PUNISHMENT OR WHAT.

All the same, a system that allowed the player a bit more specification of questions than simple keyword-use might be useful in a mystery game, for instance, where we might want to let our detective conduct inquiries

into specific details. An alternative approach to the rather free one above would be to force the player to use only questions of the form WHAT IS RAIN? or WHO ARE YOU?: this would cut down on false-positive matches. But we might still choose to store the responses in a table of this type.

Example **Chronic Hinting Syndrome** ★★★★

Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

Suppose we have a conversation system in which it is important to keep track of which subjects the player has heard mentioned. If we're careful to mark subjects in brackets, we can use the "printing the name of" activity to record which things have been mentioned so far:

```
"Chronic Hinting Syndrome"

A subject is a kind of thing.

Knowledge relates various people to various subjects.
The verb to know means the knowledge relation.

Awareness relates various people to various subjects.
The verb to be aware of means the awareness relation.

Definition: a subject is pending if the player is aware of it
and it is not known by the player.

Instead of thinking:
    if the number of pending subjects is 0, say "You have
no fresh leads at the moment.";
```

```
    otherwise say "You recall that thus far you have not
followed up with questions about [the list of pending
subjects]."


After printing the name of a subject (called idea):
    now the player is aware of the idea.
```

Now suppose that as an added convenience for the player, we let him turn on a mode in which useful conversation topics are always automatically highlighted in the text, so he doesn't waste his time trying to follow up dead leads:

```
Setting is a kind of value. The settings are bright and
dull. Understand "on" as bright. Understand "off" as dull.


Highlighting is a setting that varies. Highlighting is dull.


Understand "highlighting [setting]" as setting
highlighting. Setting highlighting is an action out of
world, applying to one setting.


Carry out setting highlighting:
    now highlighting is the setting understood.


Report setting highlighting:
    say "Highlighting is now [if highlighting is
dull]off[otherwise]on[end if]."


Before printing the name of a subject (called idea) when
highlighting is bright:
    unless the player knows the idea:
        say "[bold type]".
```

After printing the name of a subject when highlighting is bright:
    say "[roman type]".

...And the rest is peripheral.

The Sickbay is a room. "A place arranged for Nathan's comfort, since his sickness has been prolonged and because he becomes so irritating when not comfortable." The Hallway is outside from the Sickbay.

A supporter can be untried or rejected. A supporter is usually untried.

The Sickbay contains a wobbly pedestal, a table, and a sickbed. Understand "bed" as the sickbed. The pedestal, the table, and the sickbed are supporters. Nathan is a man on the sickbed. The sickbed is scenery. The initial appearance of the wobbly pedestal is "A wobbly pedestal near the door has sometimes been known to support vases of flowers, but is currently bare." The initial appearance of the table is "There is also a table of a more ordinary sort."

Nathan can be active or passive.

After printing the name of Nathan: now Nathan is passive.

Instead of putting the sculpture on the table:
    now the table is rejected;
    say "'[Not there],' [Nathan] snaps. 'The table is way too far from the sickbed.'"

Instead of putting the sculpture on the sickbed:
    now the sickbed is rejected;
    say "'[Not there],' [Nathan] rebukes you. 'You don't want me knocking it over if I roll around. In pain.'"

Instead of putting the sculpture on the pedestal:
    now the pedestal is rejected;
    say "The pedestal starts to wobble so ominously that you don't dare let go.

'[Not there],' says [Nathan]. 'That thing is falling apart.'"

Before putting something on the down: try dropping the noun instead.

To say not there:
    if all the supporters are rejected:
        say "Look, the floor would be fine";
    otherwise if the number of rejected supporters is 1:
        say "Yeah, anywhere but there, I'm afraid";
    otherwise:
        say "Come on, use your head -- I can't be watching you all the time, I'm sick".

Instead of going outside when the player is carrying the sculpture:
    say "You've still got this sculpture to get rid of."

Instead of going outside when the breakage is pending:
    say "You can't very well smash in front of [Nathan] his prize sculpture and then just scamper off without saying something. Appealing though the thought is at the moment."

Instead of going outside when a subject which is not the breakage is pending:
    say "'Yeah, go ahead,' says [Nathan], with a martyr-like air. 'It's probably best that you don't hear about [the random pending subject]. It's not something I'd go into normally.'"

The breakage is a subject. The description is "It's not a big deal. I'll just buy a new [mental wave generator].' A slight awkward pause. 'I mean, this one was a [gift], but don't worry about it". Understand "accident" or "smashing" or "breaking" or "shard" or "mishap" or "shards" or "mistake" as the breakage. Understand "sculpture" as the breakage when the player is not carrying the sculpture.

Instead of saying sorry in the presence of Nathan when the player is aware of the breakage:
    try asking Nathan about the matter of breakage.

Instead of asking Nathan to try saying sorry when the player is aware of the breakage:
    try asking Nathan about the matter of breakage.

The mental wave generator is a subject. The description is "They're kind of expensive but I can save up. I really need one, though, because of my [dreams]".

The dreams is a subject. The description is "They're not the kind of dream you want to have.' He closes his eyes and contemplates these undesirable dreams. 'Have you ever woken up convinced you were dead? No, probably not. Well... At any rate, I need the [generator]. Oh, don't worry, they're expensive but not so expensive that I won't be able to save up for another, in a few months".

The gift is a subject. The description is "[The mental wave generator] was a present from a girl named [Shari]. Actually I'm not sure she'd take to being called a girl".

Shari is a subject. The description is "Look, let's just not go into it, okay? I don't really want to relive all that right now. I still have a six-inch [scar] shaped like a banana in the middle of my back".

The scar is a subject.

Instead of asking Nathan about the matter of the scar:
    say "Nathan clears his throat, lowers his voice, and begins to tell you the story...";
    end the story saying "End of Demo -- Register to Continue!!"

Understand "ask [someone] about [any subject]" as asking it about the matter of.

Asking it about the matter of is an action applying to one thing and one visible thing.

Check asking it about the matter of:
    if the player is not aware of the second noun, say "What [second noun]?" instead;
    if the noun does not know the second noun, say "'I've no idea,' replies [the noun]." instead;
    if the player knows the second noun, say "You've already covered that. The response was '[description of the second noun].'" instead.

Carry out asking it about the matter of:
    now the player knows the second noun.

Report asking it about the matter of:
    say "'[description of the second noun],' says [the noun]."

Instead of telling Nathan about something:
    say "He pinches the bridge of his nose. 'I can't really follow this right now,' he says. 'I'm sorry.'"

Instead of asking Nathan about something:
    say "He shrugs weakly."

When play begins:
    say "'Just put that down anywhere,' says [Nathan], as you come into the room. He's sitting in the sickbed with his legs straight out in front of him. 'I don't care where.' His voice is weak, but it sharpens up for the last remark: 'And don't make a lot of noise about it.'

Considering that he woke you from a sound slumber to beg you to bring this thing over, his attitude is a bit much. You stare dubiously at the awkward crystal sculpture in your hands.";
    now Nathan knows every subject.

Instead of asking Nathan about something while the player carries the sculpture, say "[Nathan] moans dramatically and refuses to be drawn into conversation."

The player is carrying an awkward crystal sculpture. Understand "objet" or "objet de hideous" as the sculpture. The description of the sculpture is "It might

possibly be natural, or it might be man-made. It might appeal to someone, but it is certainly not to your tastes."

Instead of showing the sculpture to Nathan:
    say "'Please put it anywhere,' he says."

Instead of giving the sculpture to Nathan:
    say "'No, it doesn't work if I touch it. That's why I couldn't-- well, just put it down.'"

After dropping the sculpture:
    now the player is aware of the breakage;
    now the sculpture is nowhere;
    say "You are incredibly careful, but somehow the sculpture slips -- you might almost say slithers -- from your fingers and crashes into a thousand shards at the feet of [Nathan].

There is a tense silence."

Before reading a command: now Nathan is active.

Every turn while not asking:
    if Nathan is passive, rule succeeds;
    if the player is carrying the sculpture:
        if showing or giving, rule succeeds;
        say "[Nathan] opens one eye and stares at you meaningfully. He is waiting for you to deposit his objet de hideous somewhere.";
        rule succeeds;
    if the breakage is pending:
        if dropping, rule succeeds;
        say "You're not quite sure where to begin, but you can't very well leave without making at least some remark on the smashing of the sculpture.";
        rule succeeds;

```
    if a subject is pending:
        choose a random row in the Table of Offhand
Reminiscences;
        say "[line entry][paragraph break]".
```

```
Table of Offhand Reminiscences
line
"'It actually is kind of a funny story about [the random
pending subject],' [Nathan] remarks casually."
"[Nathan] chuckles under his breath. 'Man, I hadn't
thought about [the random pending subject] in ages.'"
"He glances sideways at you. 'It's nothing personal, you
know, but I don't feel comfortable discussing [the
random pending subject] with just anyone.'"
"'I don't know why I brought up [the random pending
subject] just now,' [Nathan] comments. 'Don't mention it
to anyone, if you don't mind.'"
"'Okay, see, the thing about [the random pending
subject] is...' [paragraph break]'Yes?' you ask, on cue.
[paragraph break]'...never mind.'"
"[Nathan] makes an explosive exasperated sound.
'Don't you want to ask me about [the random pending
subject]?' he demands."
```

```
Test me with "highlighting bright / put sculpture on
pedestal / put it on table / put it on sickbed / drop it /
think / ask nathan about breakage / think / ask him
about generator / ask him about dreams / ask him about
gift / ask him about shari / ask him about scar".
```

ExampleRobo 1

A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

```
"Robo"
```

The Experimentation Chamber is a room. Robo is a man in the Experimentation Chamber. "Robo, your prototype tin companion, stands awkwardly beside you. In the middle of his chest is a red enamel button[if the red button is switched on], currently depressed[otherwise], currently un-depressed[end if]."

The red button is a device. It is part of Robo. Instead of pushing the red button: if the red button is switched off, try switching on the red button; otherwise try switching off the red button.

After switching on the red button:
    say "CLICK! Robo is now in play-back mode."

After switching off the red button:
    say "CLACK! Robo is now in observation mode."

Definition: Robo is watching if the red button is switched off.

The current instruction set is a list of stored actions that varies.

After doing something when Robo is watching and Robo can see the player:
    now the actor is Robo;
    add the current action to the current instruction set;
    now the actor is the player;
    say "Robo watches you [the current action][one of], his yellow eyes lamp-like and observant[or]. In his metal head, gears whirr[or], his brushed-copper handlebar moustaches twitching[or] impassively[at random].";
    continue the action.

```
Every turn when Robo is not watching:
    if the number of entries in the current instruction set is
0:
        say "Robo has run out of behavior and grinds to an
(expectant) halt.";
        now the red button is switched off;
    otherwise:
        let the next task be entry 1 of the current instruction
set;
        try the next task;
        remove entry 1 from the current instruction set.


The red block and the blue cylinder are things in the
Experimentation Chamber. The counter is a supporter in
the Experimentation Chamber. The counter is scenery.


Report Robo examining Robo:
    say "Robo examines each of his hands in turn, then
each of his legs (bending over mostly double in the
middle to do this)." instead.


Report Robo examining the player:
    say "Robo stares at you, unblinkingly, for several
seconds together[if a random chance of 1 in 7
succeeds]. His left moustache-bar twitches
infinitesimally upward[end if]." instead.


Report Robo taking the cylinder:
    say "[one of][Robo] needs several attempts to get his
metal fingers around [the cylinder] -- they are not
designed for grasping small objects elegantly. But at last
he succeeds[or]Once again, Robo struggles a bit before
picking up [the cylinder][stopping]." instead.


Test me with "z / take cylinder / take block / put cylinder
on counter / put block on counter / x robo / x me / get
block / drop block / press red button / z / z / z / z / z / z /
z / z / z / z".
```

Example **Your Mother Doesn't Work Here**

Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

Suppose the player's mother is supposed to be cleaning the living room, but she can be interrupted by the need to pick up things the player has dropped. New tasks are added to the end of her "current plan" list; every turn, she attempts to do whatever is the last entry on that list.

```
"Your Mother Doesn't Work Here"

A person has a list of stored actions called the current
plan.

Every turn:
    repeat with culprit running through people who are
not the player:
        if the number of entries in current plan of the culprit
is greater than 0:
            let N be the number of entries in the current plan
of the culprit;
            try entry N of the current plan of the culprit;
            remove entry N from the current plan of the
culprit.

The Living Room is a room. It contains a somewhat
muddy Persian rug. Your mother is a woman in the
Living Room.

West of the Living Room is the Kitchen.

Instead of your mother rubbing the rug:
```

```
    say "Your mother scrubs the stained areas of the rug,
muttering to herself."


Instead of taking something:
    say "Nah, Mom'll get that."


Report your mother taking something:
    say "Your mother picks up [the noun][one of], sighing
deeply[or], jaw tight[or], with assorted comments on
your manners[or]; to judge from her comments, she is
also indulging in a pleasant fantasy about Swiss
boarding schools[stopping]." instead.


When play begins:
    add mother going west to the current plan of mother;
    add mother rubbing the rug to the current plan of
mother.


Every turn:
    if mother is not in the Living Room, end the story
finally.


Carry out dropping something:
    add mother taking the noun to the current plan of
mother.


The player carries some dirty socks, some dirty shoes, a
dirty hat, a pair of dirty trousers, and a backpack.


Test me with "drop socks / z / drop shoes / drop hat /
drop all / z / z".
```

As goal-seeking goes, this is fairly rudimentary; "Boston Cream" provides an alternative (and slightly more sophisticated approach), but for really

complex goal-seeking characters, it is probably best to turn to the character extensions designed for Inform.

ExamplePine 2 *WI*

Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

```
"Pine"

A person can be asleep or awake. A person can be
active or passive.

The Spinning Tower is a room. "A remote corner of the
old castle, reserved for spinning and weaving tasks."

Sleeping Beauty is an asleep woman in the Spinning
Tower. "[if asleep]Sleeping Beauty lies here, oblivious to
your presence[otherwise]Sleeping Beauty stands beside
you, looking a little confused[end if]." The description is
"She is even more magnificent than the rumors
suggested." Understand "woman" or "girl" or "princess"
or "lady" as Sleeping Beauty.

Discovery is a scene. Discovery begins when play
begins. Discovery ends when Sleeping Beauty is
awake. Marriage Proposal is a scene. Marriage
Proposal begins when Discovery ends.

When Discovery ends: say "Throughout the palace you
can hear the other sounds of stirring and movement as
the spell of centuries is broken."
```

```
Instead of waking an awake person: say "Redundant."


Instead of waking an asleep person: say "Yes, but
how?"


Instead of attacking an asleep person:
    now the noun is awake;
    say "[The noun] sits bolt upright. 'Hey! Ow!' So much
for that true love's kiss nonsense."


Instead of kissing an asleep person:
    now the noun is awake;
    say "[The noun] slowly stirs to wakefulness!"


Instead of throwing water at an asleep person:
    now the second noun is awake;
    now the noun is nowhere;
    say "You pour out [the noun] on [the second noun].


[The second noun] wakes, shuddering. 'Agh! I had a
terrible dream about drowning and then-- Hey!'"


The player carries a jug of water. Understand "pour
[something] on [something]" or "splash [something] at/
on [something]" as throwing it at.
```

So much, we had before. Now, suppose we want a conversation style which allows the player to move conversation forward by asking appropriate questions, but which will keep moving forward even if he doesn't. To this end, we provide a table -- a borrowing from a later chapter. In the table, we record two ways of performing each conversation bit, one which reflects the player's participation, and one in which the character moves things onward:

Table of Conversation

"Sleep well? You ask solicitously.

"Oh really, I'm perfectly happy, thanks. You?"

"Not so much. Some of the beds are too short for that angle."

"Sorry to hear that. This is a little embarrassing, I just met, but, um. Would you like to get married?'

She looks at you. 'No. Do I look like I want to marry you or something.'"

"I was told I was going to marry the ruler of the kingdom,' you say, apologetically. 'Would that be very bad?'

'Oh, it's not you -- I'm seeing someone,' she says, smiling quickly.

"Do you think I could go look for him? I'm a searching type, since I last saw him a hundred years ago."

"You've been updated for modern sensibilities, I see," you say. An unpleasant thought occurs to you. 'Was your young man in the castle somewhere?'

She looks to the window and looks out at the now-fading thicket of briar. 'That took a while to grow,' she observes. 'I've been up here longer than I thought.'

You shrug, uncomfortable."

```
Instead of asking an awake beauty about a topic listed
in the Table of Conversation:
    now Beauty is passive;
    say "[reply entry][paragraph break]";
    blank out the whole row.
```

The "now Beauty is passive" line prevents her from making any conversation of her own on a turn when we've spoken to her. This keeps the conversation from progressing too quickly.

```
Instead of telling an awake beauty about something: try
asking the noun about it.
```

```
Instead of asking an asleep person about something:
    say "[The noun] snores."
```

```
Marriage Proposal ends when the number of filled rows
in the Table of Conversation is 0.
```

```
Every turn during Marriage Proposal:
    if Beauty is active:
        repeat through Table of Conversation:
            say "[quip entry][paragraph break]";
            blank out the whole row;
            make no decision.
```

After we've generated any spontaneous conversation, we return her to her regular active state.

Every turn: now Beauty is active.

When Marriage Proposal ends: end the story saying "This is going to take some explaining."

Test me with "x beauty / wake beauty / pour water on beauty / ask beauty about sleep / z / ask beauty about marriage".

Now we have a scenario in which the player can ask her some questions out of order if he really wants to, but the scene will not end until the basic conversation topics have been exhausted. If we wanted to add some other chit-chat, not as part of the main conversation strand, but by way of optional enrichment, we might make a second conversation table and record alternative outcomes in it.

ExampleRobo 2

A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

We have seen how we can make a robot that watches the player, then plays back the same actions again. A slightly more adventurous implementation would be to let the player create a whole series of named scripts which the robot will run on command.

To do this, we'll need each program to have a command that sets it off (stored as text, since this is the best way to capture and preserve arbitrary text entered by the player) and then the script of actions that must result:

"Robo 2"

Use dynamic memory allocation of at least 16384.

```
Chapter 1 - Programming

Section 1 - The Programs Themselves

The hard drive is a container. A program is a kind of
thing. 15 programs are in hard drive. A program has
some text called the starter command. A program has a
list of stored actions called the script. Understand the
starter command property as describing a program.

Rule for printing the name of a program (called the
target) which is not blank:
    say "[starter command of the target in upper case]".

Definition: a program is blank if the number of entries in
its script is 0.

The current instruction name is some text that varies.
The current instruction set is a list of stored actions that
varies.
```

Now, we want to let Robo learn new programs; for this purpose, we'll emulate the code from our previous implementation, so that Robo watches what the player does and stores those actions in his script:

```
Section 2 - Learning New Programs

Understand "learn [text]" as learning. Learning is an
action applying to one topic.
```

```
Check learning:
    say "You have already learned all you need to know.
Robo, however, remains to be trained." instead.


Check Robo learning:
    if Robo is watching, say "Robo is already recording
'[the current instruction name]'." instead.


Carry out Robo learning:
    truncate the current instruction set to 0 entries;
    now the current instruction name is the topic
understood;
    now Robo is watching.


Report Robo learning:
    say "'Learning [the current instruction name in upper
case],' Robo replies."


After doing something when Robo is watching and Robo
can see the player:
    now the actor is Robo;
    add the current action to the current instruction set;
    now the actor is the player;
    say "Robo watches you [the current action][one of],
his yellow eyes lamp-like and observant[or]. In his metal
head, gears whirr[or], his brushed-copper handlebar
moustaches twitching[or] impassively[at random].";
    continue the action.
```

Of course, we also need to be able to switch learning mode off, and store any script learned this way. We'll also use the same STOP command to make Robo terminate a program he's in the middle of running.

```
Section 3 - Returning to Standby Mode


Understand "stop" as stopping. Stopping is an action
```

```
applying to nothing.


Check stopping:
    say "The command is useful only for Robo." instead.


Check Robo stopping:
    if Robo is standing by, stop the action.


Carry out Robo stopping when Robo is watching:
    let N be a random blank program;
    if N is a program:
        now the starter command of N is the current
instruction name;
        now the script of N is the current instruction set;
        say "'Stored [current instruction name in upper
case].'";
    otherwise:
        say "FAILURE: no program slots remaining."


Carry out Robo stopping:
    now Robo is standing by.


Report Robo stopping:
    say "Robo is now standing by."
```

Next, we need to be able to play these programs back again. We'll give Robo a "current program" to store which program he's currently working on, and a number called "stage" which will record where he is in the script. Our previous implementation simply had Robo erase entries from his script list as he performed them, but this time we would like Robo to be able to remember and rerun the same scripts over and over, so we need something a little more subtle.

Section 4 - Running Learned Programs

```
Understand "run [any program]" as running. Running is
an action applying to one visible thing.



Check running:
    say "Only Robo can perform Robo's programs."
instead.



Check Robo running:
    if Robo is not standing by, stop the action.



Unsuccessful attempt by Robo running:
    say "'ERROR: Robo can launch new programs only
when on standby.'"



Carry out Robo running:
    now the current program of Robo is the noun;
    now the stage of Robo is 1;
    now Robo is replaying.



Report Robo running:
    say "'Running [the starter command in upper case],'
Robo confirms."



Every turn when Robo is replaying:
    let the chosen script be the script of the current
program of Robo;
    let maximum be the number of entries in the chosen
script;
    let N be the stage of Robo;
    let the next step be entry N of the chosen script;
    try the next step;
    increment the stage of Robo;
    if the stage of Robo is greater than the maximum:
        say "Robo's program ends, and he reverts to
stand-by mode.";
        now Robo is standing by;
```

```
now the stage of Robo is 1.
```

For the player's sanity, we should also provide a way to find out which programs Robo has stored in memory and what they do, so we design two listing commands:

```
Section 5 - Listing Learned Programs

Understand "list programs" as requesting program list.
Requesting program list is an action applying to nothing.

Check requesting program list:
    say "You will have to ask Robo to list programs."
instead.

Carry out Robo requesting program list:
    say "'The available program[if more than one program
is not blank]s[end if] [is-are list of programs which are
not blank].'".

Understand "describe [any program]" or "list [any
program]" as requesting script of. Requesting script of is
an action applying to one visible thing.

Check requesting script of:
    say "You will have to ask Robo to give you the script."
instead.

Carry out Robo requesting script of:
    say "The script of [noun] is: [script of the noun]."
```

And to complete the suite, in case the player runs into Robo's fifteen-program limit:

```
Section 6 - Deleting Learned Programs

Understand "delete [any program]" as deleting. Deleting
is an action applying to one visible thing. Understand
the command "erase" as "delete".


Check deleting:
    say "You will have to instruct Robo to delete [the
noun]." instead.


Check Robo deleting (this is the can't delete except in
standby rule):
    if Robo is not standing by, stop the action.


Unsuccessful attempt by Robo deleting:
    say "'ERROR: programs may only be deleted while
Robo is in stand-by mode.'" instead.


Carry out Robo deleting:
    truncate the script of the noun to 0 entries;
    now the starter command of the noun is "".


Report Robo deleting:
    say "'Program deleted.'"
```

Now we use pretty much the same set-up as before to test Robo's abilities:

```
Chapter 2 - The Scenario
```

The Experimentation Chamber is a room.

Robo is a man in the Experimentation Chamber. "Robo, your prototype tin companion, stands awkwardly beside you[if watching], watching[end if]." Robo can be watching, replaying, or standing by. Robo is standing by. Robo has a program called the current program. Robo has a number called the stage.

Persuasion rule: persuasion succeeds.

The red block and the blue cylinder are things in the Experimentation Chamber. The counter is a supporter in the Experimentation Chamber. The counter is scenery.

Report Robo examining Robo:
    say "Robo examines each of his hands in turn, then each of his legs (bending over mostly double in the middle to do this)." instead.

Report Robo examining the player:
    say "Robo stares at you, unblinkingly, for several seconds together[if a random chance of 1 in 7 succeeds]. His left moustache-bar twitches infinitesimally upward[end if]." instead.

Report Robo taking the cylinder:
    say "[one of][Robo] needs several attempts to get his metal fingers around [the cylinder] -- they are not designed for grasping small objects elegantly. But at last he succeeds[or]Once again, Robo struggles a bit before picking up [the cylinder][stopping]." instead.

Test me with "test chocolate / test vanilla".

```
Test chocolate with "learn chocolate / stop / list
programs / Robo, learn chocolate / take red / put all on
counter / Robo, stop / Robo, list programs / Robo, run
chocolate / z / Robo, run chocolate / Robo, stop / z".
```

```
Test vanilla with "Robo, learn vanilla / take all / i / drop
all / x robo / x me / Robo, stop / Robo, list programs /
Robo, list vanilla / Robo, run vanilla / z / z / robo, delete
vanilla / robo, stop / robo, list vanilla / robo, delete
vanilla / robo, list programs".
```

We could also have written this so that Robo learns new scripts by accepting the player's instructions, so that

```
>ROBO, LEARN LIBRARY THEFT
>ROBO, TAKE BOOK
>ROBO, EAST
>ROBO, STOP
```

...would store the script 'library theft' with the actions taking the book and going east. The implementation there would have been mostly identical, except that instead of an "after doing something..." rule, we would have captured commands as we asked Robo to perform them, and added those to the command list in progress. The alternative code might look something like this:

```
Before Robo doing something other than stopping when
Robo is watching:
    add the current action to the current instruction set;
    say "'CHECK: [current action] added to script,' says
Robo." instead.
```

```
Unsuccessful attempt by Robo doing something when
Robo is watching:
    say "He does not actually perform the action."
```

Example **Mistress of Animals**

A person who moves randomly between rooms of the map.

Suppose we want a restless sort of character always pacing from room to room. It is quite easy to use adjacency to achieve this effect:

```
"Mistress of Animals"

Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.

Artemis is a woman in Thebes.

Every turn:
    if Artemis is in a room (called the current space):
        let next space be a random room which is adjacent
to the current space;
        if Artemis is visible, say "Artemis heads to [the next
space].";
        move Artemis to next space;
        if Artemis is visible, say "Artemis arrives from [the
current space]."

Test me with "z / z / z / z / z / z".
```

Of course, it helps that Artemis is the sort to like open spaces. The implementation would become more complicated if there were doors which might block transit between these locations.

Suppose we want to write a character who tries to be in the same room as the player. We will do this by testing every turn whether the character's location and the player's location are the same; if the answer is no, the character will look for a path to the player's location, then try to move along that path. (We will learn more about finding paths and giving characters instructions later.)

The result will be that if the player ever moves to another location, the character will automatically pursue him.

```
"Van Helsing"


The Drawbridge is a room. North of the Drawbridge is
the Immensely Enormous Entry Hall. West of the Entry
Hall is the Vast Dining Area. North of the Vast Dining
Area is the Colossal Kitchen. The Spooky Guano-filled
Attic is above the Entry Hall.


Count Dracula is a man in the Attic.
```

In the following condition, we could also have written "if the location of Count Dracula is not the location", because "location" by itself is always understood to be the player's location. But it seemed better for clarity to write it this way:

```
Every turn:
    if the location of Count Dracula is not the location of
the player:
        let the way be the best route from the location of
Count Dracula to the location of the player, using doors;
        try Count Dracula going the way;
    otherwise:
        say "'Muhahaha,' says Count Dracula."
```

Test me with "z / z / n / w / e / u / z / d".

 ExampleOdyssey *WI*

A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

```
"Odyssey"

Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.

Athena is a woman in Athens.
```

Athena will proceed, unless delayed, through a list of locations stored in a simple table. Rather than using Inform's route-finding abilities ("the best route from..."), we simply move Athena from one location to the next, not even using the going action: she moves in mysterious ways, as befits a goddess.

```
Table of Athena's Movement
destination
Thebes
Delphi
Thebes
Athens
Corinth
Mycenae

Every turn when Athena is active:
    repeat through the Table of Athena's Movement:
```

```
        let last space be the location of Athena;
        if Athena can be seen by the player, say "Athena
    heads to [the destination entry].";
        move Athena to destination entry;
        if Athena can be seen by the player, say "Athena
    arrives from [the last space].";
        blank out the whole row;
        break.
```

By blanking out the table line by line, we make sure that we never lose our place in the path.

Since we want the player to be able to talk to Athena, we need a way to stall her in her path, as well.

```
Athena can be active or passive. Athena is active.


Before doing something to Athena:
    now Athena is passive;
    say "Athena waits around patiently, though you can
tell she would like to leave..."


Instead of telling Athena about something:
    say "She watches you patiently as though to say that
she already knows."


Instead of asking Athena about something:
    say "Her response is inscrutably ancient and Greek.
Afterwards you remember only the flash of bright eyes."
```

Finally, we do need to wake Athena up again if she has become passive. The following rule will occur after the movement rule just because of code ordering, though we could make matters more explicit if we needed to:

```
Every turn when Athena is passive:
    now Athena is active.
```

Test me with "east / northwest / wait / examine athena / wait".

Example Actaeon

302

A FOLLOW command allowing the player to pursue a person who has just left the room.

WI

Suppose we want the player to be able to go after characters who are moving around the map. The trick, of course, is that once characters are gone they are no longer visible to "follow [person]", so we need "follow [any person]" to find them.

"Actaeon"

A person has a room called last location.

Understand "follow [any person]" as following.
Understand the commands "chase" and "pursue" as "follow".

Following is an action applying to one visible thing.

Check following:
    if the noun is the player, say "Wherever you go, there you are." instead;
    if the noun is visible, say "[The noun] is right here." instead;
    if the last location of the noun is not the location, say "It's not clear where [the noun] has gone." instead.

Here again the best route comes in handy:

```
Carry out following:
    let the destination be the location of the noun;
    if the destination is not a room, say "[The noun] isn't
anywhere you can follow." instead;
    let aim be the best route from the location to the
destination;
    say "(heading [aim])[line break]";
    try going aim.
```

```
Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.
```

```
Artemis is a woman in Corinth.
```

We do also have to make sure that whenever we move a person from room to room, we record where they were moved from; otherwise, our clever restrictions about whom the player can pursue will not work properly.

```
To move (pawn - a person) tidily to (target - a room):
    now the last location of the pawn is the holder of the
pawn;
    move the pawn to the target.
```

```
Every turn:
    let current location be the location of Artemis;
    let next location be a random room which is adjacent
to the current location;
    if Artemis is visible, say "Artemis heads to [the next
location].";
    move Artemis tidily to next location;
    if Artemis is visible, say "Artemis arrives from [the
current location]."
```

Test me with "wait / follow artemis / follow artemis / follow artemis".

ExampleLatris Theon

A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

To begin with, we create an action for going to a named place. All that this action will do is to change that person's hoped-for destination: the actual moving around comes later.

```
"Latris Theon"

A person has a room called destination.

Understand "go to [any room]" as going vaguely.

Going vaguely is an action applying to one visible thing.

Carry out someone going vaguely:
    now the destination of the person asked is the noun.

Report someone going vaguely:
    say "[The person asked] looks amused, but accepts
the commission to go to [the noun]."
```

It stands to reason the player plays Zeus or at the very least Apollo, but let's not let this go to the player's head. Note that the following rule applies to the player, but not to anyone else, so HERMES, GO TO

ATHENS will work but GO TO ATHENS will not.

```
Carry out going vaguely:
    say "You're too thoroughly lost."
```

And finally we recreate Greece and one of its heroes.

```
Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.
```

```
Hermes is a man in Corinth. The destination of Hermes
is Corinth. [So he is initially happy where he is.]
Persuasion rule for asking Hermes to try going vaguely:
persuasion succeeds. [But he is open to suggestions.]
```

```
Every turn when the destination of Hermes is not the
location of Hermes:
    let the right direction be the best route from the
location of Hermes to the destination of Hermes;
    try Hermes going the right direction.
```

```
Test me with "hermes, go to athens / e".
```

It simplifies matters that our map of Greece makes it possible to reach any location from any other location, by some sequence of movements: if there were an isolated location -- say, Crete -- with no map connection to the mainland, then we would have to worry about the "right direction" not being a direction at all. The following version of Hermes' trekking rule is protected against the possibility:

```
Every turn when the destination of Hermes is not the
location of Hermes:
```

let the right direction be the best route from the location of Hermes to the destination of Hermes;
if the right direction is a direction, try Hermes going the right direction.

---

402 **ExamplePatient Zero** *WI*

People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

"Patient Zero"

Use the serial comma and no scoring.

Understand "about" as asking for information. Asking for information is an action out of world.

Carry out asking for information: say "An implementation of the following creative brief:

People wander around some small map, on errands. One, sad to tell, has Gelato's Syndrome, a tragic condition turning one's skin the colour of a random flavour of ice cream (raspberry ripple, neapolitan, etc.). When two people are in the same room, there's a 1/3 chance that an infected person will infect a non-infected one. The player can cure any single person: victory condition - to stamp out the disease."

When play begins:
say "Gelato's Syndrome. It's struck, and it's struck hard. In these sticky summer months, there's no telling

who will contract the disease next.";
    now the command prompt is "[if the destination of the player is not blank](heading to [destination of the player]) [end if]>".

Section 1 - Errands

The current actor is a person which varies. The current owner is a person which varies.

Every turn:
    if player is active, follow the character movement rules.

Every turn:
    now the last person named is the player;
    now the last thing named is the player;
    now every person is active.

A person can be active or passive. The player is passive.

The character movement rules are a rulebook.

The first character movement rule:
    now group size is 1;
    now the last person named is the player;
    now the last thing named is the player;
    now the player is passive.

A character movement rule:
    repeat with mover running through innocent people:

```
        now the current actor is the mover;
        follow the shopper rules;
        now the current actor is passive;
    follow the movement reporting rule.


A character movement rule:
    repeat with next mover running through mercantile
people:
        now the current owner is the next mover;
        follow the shopowner rules;
        now the current owner is passive;
    follow the infection rule.


To decide whether movement has not yet occurred:
    if the player is passive, no;
    yes.


Definition: a person is mercantile if it owns a room.
Definition: a person is innocent if it is not mercantile and
it is not the player.


The shopowner rules is a rulebook.


A shopowner rule:
    let the shop be a random room owned by the current
owner;
    if the shop is air-conditioned and an open door (called
the escape) protects the shop, try the current owner
closing the escape instead.


Report someone closing a door when the person asked
owns the location:
    say "[The person asked], muttering darkly about air-
conditioning and electricity, closes [the noun]." instead.
```

Report Vanessa closing the metal door when the metal door is visible:
    if Vanessa is visible, say "Vanessa watches serenely as the metal door slides automatically back in place, sealing Cold Comfort." instead;
    otherwise say "The metal door slides heavily back into place." instead.

A shopowner rule:
    if the location of the current owner encloses a submitted artwork (called the target):
        try the current owner filing the target.

Filing is an action applying to one thing.

Before someone filing something which is not carried by the person asked:
    try the person asked taking the noun instead.

Carry out someone filing:
    if the person asked does not carry the noun and the person asked is visible, say "[The person asked] tries unsuccessfully to get [the noun]." instead;
    now the noun is nowhere.

Report someone filing:
    say "[The person asked] registers [the noun] and files it away."

The shopper rules is a rulebook.

A shopper rule:
    if the current actor carries something (called the problem), try the current actor resolving the problem

instead.

A shopper rule:
    if the current actor is not in the pool hall and the air conditioner is switched on:
        try the current actor approaching the pool hall;
    otherwise:
        let way be a random direction;
        try the current actor going the way.

Definition: a room is air-conditioned:
    if it is outdoors, no;
    if it is the Pool Hall and the air conditioner is switched off, no;
    if it is protected by a door, yes;
    no.

Protection relates a door (called X) to a room (called Y) when the front side of X is Y or the back side of X is Y. The verb to protect means the protection relation.

Ownership relates one person to various rooms. The verb to own means the ownership relation.

Resolving is an action applying to one thing.

An artwork is a kind of thing. Before printing the name of an artwork, say italic type. After printing the name of an artwork, say roman type. An artwork can be submitted or reserved.

A book is a kind of artwork.

Before someone resolving a book when the person asked is not in the Public Library:
    try the person asked approaching the Public Library instead.

Carry out someone resolving a book:
    move the noun to the Public Library;
    now the noun is submitted.

Report someone resolving a book:
    say "[The person asked] turns in [the noun]."

Before listing contents: group books together.

Before grouping together books: say "books entitled ".

A stamped envelope is a kind of thing.

Before someone resolving a stamped envelope when the person asked is not in the Post Office:
    try the person asked approaching the Post Office instead.

Carry out someone resolving a stamped envelope:
    now the noun is nowhere.

Report someone resolving a stamped envelope:
    say "[The person asked] slips [a noun] into the outgoing mail slot."

```
Instead of someone resolving a stamped envelope
when the person asked carries at least two stamped
envelopes:
    if the person asked is visible, say "[The person asked]
shoves into the mail slot [a list of stamped envelopes
carried by the person asked].";
    repeat with item running through stamped envelopes
carried by the person asked:
        now the item is nowhere.



A DVD is a kind of artwork.



Before someone resolving a DVD when the person
asked is not in the Rental Store:
    try the person asked approaching the Rental Store
instead.



Carry out someone resolving a DVD:
    now the noun is submitted;
    move the noun to the Movie Rental Store.



Report someone resolving a DVD:
    say "[The person asked] returns [the noun]."



Instead of someone resolving a DVD when the person
asked carries at least two DVDs:
    if the person asked is visible, say "[The person asked]
turns in [a list of DVDs carried by the person asked].";
    now every DVD carried by the person asked is
submitted;
    now every DVD carried by the person asked is in the
location of the person asked.



Before listing contents: group DVDs together.
```

Before grouping together DVDs: say "DVDs of ".

Approaching is an action applying to one thing.

Carry out someone approaching:
    let the way be the best route from the location of the person asked to the noun, using doors;
    if the way is a direction, try the person asked going the way;
    otherwise stop the action.

A coupon is a kind of thing.

Carry out someone resolving a coupon:
    try the person asked giving the noun to Vanessa.

The block giving rule is not listed in any rulebook.

Check giving something to someone (this is the block player giving rule):
    abide by the block giving rule.

Before someone resolving a coupon when the person asked is not in Cold Comfort:
    try the person asked approaching Cold Comfort instead.

After someone giving a coupon to Vanessa:
    let the reward be a random ice cream cone;
    let the new flavor be a random infection color;
    now the infection color of the reward is the new flavor;
    move the reward to the person asked;

```
    now the noun is nowhere;
    if Vanessa is visible, say "[The person asked] trades
in [the noun] and receives [a reward] from Vanessa."


Infection color is a kind of value. The infection colors are
french vanilla, whole-bean vanilla, mint, chocolate, dark
chocolate, chocolate chip, chocolate fudge, mint
chocolate chip, chocolate chocolate chip, triple
chocolate, white chocolate, white chocolate chip, aztec
cocoa-chili, raspberry ripple, neapolitan, rum raisin,
dulce de leche, strawberry chunk, rocky road,
blackberry sorbet, lemon sherbet, lime ice, caramel
swirl, mango, saffron silk, and cookie dough cream.


To say list of flavors:
    let current color be french vanilla;
    while current color is not cookie dough cream:
        say "[current color], ";
        now current color is the infection color after the
current color;
    say "and [current color]".


Understand "ask vanessa for [flavored ice cream]" as
buying the flavor. Understand "buy [flavored ice cream]"
as buying the flavor.


Buying the flavor is an action applying to one infection
color.


Check buying the flavor:
    unless the player can see Vanessa:
        say "It would help if you were in the presence of an
ice cream salesperson." instead.


Carry out buying the flavor: say "'Do you have a
coupon?' Vanessa demands. You admit you do not. 'No
```

[infection color understood] for you!"

Understand "ice cream" or "cream" or "ice" or "sherbet" or "sorbet" as "[ice cream]".

Understand "[infection color]" or "[infection color] [ice cream]" as "[flavored ice cream]".

An ice cream cone is a kind of thing. An ice cream cone is always edible. An ice cream cone has an infection color. An ice cream cone can be half-eaten or fresh. Understand the infection color property as referring to an ice cream cone.

Carry out someone resolving an ice cream cone:
    try the person asked eating the noun instead.

Instead of someone eating a fresh ice cream cone:
    now the noun is half-eaten;
    if the person asked is visible, say "[The person asked] licks [the noun]."

Report someone eating an ice cream cone:
    say "[The person asked] pops the end of [the noun] into [if the person asked is female]her[otherwise]his[end if] mouth and swallows." instead.

Before printing the name of an ice cream cone:
    say "[if half-eaten]half-eaten [end if][infection color] ".

Section 2 - Infection Rules

This is the infection rule:
    if an infected person (called typhoid mary) can see a clean person (called random bystander) and a random chance of 1 in 3 succeeds:
        try typhoid mary sneezing on the random bystander.

A person can be infected or clean. A person has an infection color.

Every turn:
    if the player is infected, say "You feel itchy."

Definition: a person is other if it is not the player.
Definition: a person is another if it is other.

When play begins: now right hand status line is "Sick: [number of infected people]/[number of people]".

Every turn:
    if every person is infected, end the story saying "Everyone succumbs";
    if every person is clean, end the story finally saying "The Syndrome is eradicated".

Understand "sneeze on [something]" as sneezing on.
Sneezing on is an action applying to one thing.

Check sneezing on:
    if the player is clean, say "You're not sickly." instead;
    if the noun is the player, say "Ew." instead;
    if the noun is not a person, say "[The noun] cannot be

```
infected." instead.


Carry out sneezing on:
    now the noun is infected;
    now the infection color of the noun is a random
infection color.


Carry out someone sneezing on:
    now the noun is infected;
    now the infection color of the noun is a random
infection color.


Report sneezing on:
    say "Unable to control yourself, you sneeze on
[noun].".


Report someone sneezing on:
    say "[The person asked] sneezes on [if the noun is
the player]you[otherwise][noun][end if]!".


Understand "inject [someone] with [something]" as
injecting it with. Understand "inject [someone] with
[syringe]" as injecting it with. Understand "use [syringe]
on [someone]" as injecting it with. Understand the
commands "innoculate" and "vaccinate" as "inject".


Injecting it with is an action applying to two things.


Check injecting it with:
    if the second noun is not the syringe, say "[The
second noun] cannot inject anything." instead;
    if the noun is clean:
        if the noun is the player, say "You're not infected
yet." instead;
```

say "[The noun] is not infected, and the syringe contains a cure, not a vaccine." instead.

Carry out injecting it with: now the noun is clean.

After injecting the player with something: say "You inject yourself, wincing at the sting. But the itching fades almost at once."

Report injecting it with: say "You inject [the noun], who is now cured (but could easily be reinfected)."

Section 3 - Geography

Include Locksmith by Emily Short.

Understand "go to/toward/into [any room]" as going toward. Understand "enter [any room]" as going toward.

A person has a room called the destination.

Going toward is an action applying to one thing.

Check going toward:
    if the noun is the location, say "You're already in [the location]." instead.

Carry out going toward:
    now the destination of the player is the noun;
    let heading be the best route from the location to the noun, using even locked doors;
    if heading is not a direction, say "You can't think how to get there from here." instead;
    try going heading;
    if the location is the destination of the player, now the destination of the player is blank.

Instead of waiting when the destination of the player is not blank:
    if the destination of the player is the location:
        now the destination of the player is blank;
    otherwise:
        try going toward destination of the player;
        if the location is the destination of the player, now the destination of the player is blank.

Understand "stop" or "cease" as stopping. Stopping is an action applying to nothing. Carry out stopping: now the destination of the player is blank. Report stopping: say "You stop in your tracks."

After going to an air-conditioned room:
    say "You step into the mercifully air-conditioned surroundings of...";
    continue the action.

After going from an air-conditioned room:
    say "You emerge from the air-conditioning into heat like a wall...";
    continue the action.

Instead of listening to an air-conditioned room:
    say "The air-conditioning hums softly."

The Alfred Cralle Pool Hall is a room. "The town's most popular gathering-place, the pool hall is decorated in honor of the inventor of the ice cream scoop." The air conditioner is a device in the Pool Hall. "[if switched off]An air conditioner sits in the corner, unhappily inert[otherwise]The air conditioner hums briskly[end if]."

The felt door is west of the Pool Hall. The felt door is a door. The felt door is open. The felt door is lockable and unlocked. The key to the city unlocks the felt door. The description of the felt door is "It has a prominent lock, designed for an old-fashioned key."

After locking a door with something in the presence of an other person (called audience):
    say "[The audience] looks a little non-plussed when you lock [the noun], but shrugs."

Nancy Johnson Memorial Square is west of the felt door. The description of Nancy Johnson Memorial Square is "Waves of August heat rise from the pavement: more than once you've had the fancy that your shoes are simply going to stick. At the center of the square, rubbed to a brownish polish by many adoring hands, is the statue of Mrs. Nancy Johnson of New Jersey."

The statue is scenery in Memorial Square. Understand "nancy" or "johnson" or "mrs" as the statue. The description of the statue is "Mrs. Johnson is pictured with a hand-cranked ice cream freezer tucked under one arm. Her other hand grips an ice cream scoop, ready to serve frozen dessert to the huddled masses." A hand-cranked ice cream freezer is part of the statue. The description is "The hand-cranked ice cream freezer was Mrs. Johnson's invention in 1846, though it was William Young who had the sense to patent the thing in 1848." The scoop is part of the statue. The description of the scoop is "An anachronism: Alfred Cralle would not invent the tool until 1897."

The Post Office is northwest of Nancy Johnson Memorial Square. "Service at the post office is on the slow side since everything went automated." The slot is scenery in the post office. The slot is a container. Carry out inserting something into the slot: now the noun is nowhere. Report inserting something into the slot: say "[The noun] falls out of sight, and you know you will never see it again."

Hamwi Street is northeast of an iron gate. "A U-shaped street running from Main Street around to the Memorial Square, Hamwi Street was recently added by ambitious city planners. The small and straggly line of trees has yet to grow enough to provide perceptible shade, so the street is even hotter and more unforgiving than the other parts of town."

The iron gate is northeast of Nancy Johnson Memorial Square. The iron gate is a door. It is lockable and unlocked.

Before printing the name of the iron gate while not opening or closing or locking or unlocking:
    if the person asked is the player:
        if the gate is open, say "open ";
    otherwise if the gate is locked:
        say "locked ";
    otherwise if the gate is closed:
        say "closed ".

Cold Comfort Ice Cream is north of a metal door. The metal door is north of Hamwi Street. A poster is fixed in place in Cold Comfort. "A poster fills one wall with the blazing promise of treats to come." The description of the poster is "Coming soon! Thai ice creams! Durian, jackfruit, taro, and coconut flavors!"

The metal door is a door. "A frosty metallic door separates [the location] from [the other side of the metal door]." The metal door is lockable and unlocked. The key to the city unlocks the metal door.

Marciony Street is southeast of Nancy Johnson Memorial Square. "A semi-circular terrace, named somewhat fancifully after one claimant to the invention of the ice cream cone -- though Hamwi Street competes for the same honor. There are wedges of cool shadow here and there thanks to the buildings, but for the most part the southern exposure keeps Marciony unpleasantly hot."

The Movie Rental Store is west of a glass door. The glass door is a door. It is west of Marciony Street. The glass door is lockable and unlocked. The key to the city unlocks the glass door.

Main Street is southeast of Hamwi Street. Main Street is northeast of some bronze gates.

The emergency box is in Main Street. The emergency box is fixed in place. "A fire-red box with a glass front faces the sidewalk, with 'In case of emergency, BREAK GLASS' lettered on it." The emergency box is closed and transparent. Understand "glass" as the box. Instead of attacking the closed emergency box: say "You hit the emergency box, which shatters open."; now the emergency box is open. Instead of attacking the open emergency box: say "The glass has already been thoroughly broken."

The syringe is in the emergency box. The description of the syringe is "It contains the cure for Gelato's Syndrome. You can inject anyone you like with it."

The bronze gates are northeast of Marciony Street. The bronze gates are a door. The bronze gates are lockable and unlocked. The description of the bronze gates is "Erected during the milk-taint revolution of 1937, they were designed to keep Main Street safe from the depredations of dairy-starved rioters."

The Public Library is east of Main Street. "Built in the 1920s during the height of the dairy boom, the public library has lush pink velvet seats, marble walls the color of fresh cream, and a motif of cherries carved around every doorframe. An incongruous sign hangs from the ceiling." The incongruous sign is scenery in the Public Library. The description of the incongruous sign is "Eating and drinking in the library is STRICTLY PROHIBITED."

Town Hall is southeast of Main Street. "Town Hall was built during the slow days of the ice-cream bust, and therefore it is as joyless and utilitarian as the Public Library is ridiculous. Unwilling to be reminded of their pain, the inhabitants steered clear of any decoration that might remotely be construed to resemble a scoop of anything: so there are no curves, only disciplined right angles." The key to the city is in Town Hall. It unlocks the iron gate. It unlocks the bronze gates. The description of the key to the city is "A skeleton key."

A room can be indoors or outdoors. The Post Office, the Alfred Cralle Pool Hall, the Store, Cold Comfort, Town Hall, and the Library are indoors.

Use full-length room descriptions.

After looking in an outdoors room:
    let started printing be false;
    now every proximate door is not mentioned;
    if an indoors room is adjacent:
        let started printing be true;

```
        say "From here you can head into [the list of
adjacent indoors rooms][if a proximate door is not
mentioned], or go through [the list of proximate doors
which are not mentioned][end if]. [run paragraph on]";
    if an outdoors room is adjacent:
        say "You could[if started printing is true] also[end if]
go ";
        let count be the number of adjacent outdoors
rooms;
        let index be count;
        repeat with next room running through adjacent
outdoors rooms:
            let way be the best route from the location to the
next room;
            say "[way] to [the next room]";
            decrement index;
        make delimiter index of count, continuing;
    if a proximate door is not mentioned:
        let started printing be true;
        say "[if started printing is true]Also
available[otherwise]Your available exits[end if] [is-are
the list of proximate doors which are not mentioned].";
    otherwise:
        if started printing is true, say paragraph break.



Definition: a door is proximate:
    if the front side of it is the location, yes;
    if the back side of it is the location, yes;
    no.



Before exiting when the player is in an indoors room:
    if the player can see a door (called nearest exit), try
entering the nearest exit instead;
    repeat with way running through directions:
        let next room be the room way from the location;
        if the next room is a room, try going way instead.



Blank is a room. The destination of the player is Blank.
Blank contains 15 ice cream cones.
```

Section 4 - Other Players

Vanessa is a woman in Cold Comfort. Vanessa owns Cold Comfort.

Francine is a woman in the Public Library. Francine carries a book called Phlox for Phyllis. Francine carries a stamped envelope called a pink stamped envelope.

Lewis is a man in the Alfred Cralle Pool Hall. Lewis carries 3 stamped envelopes. Lewis carries a book called Idiot's Guide to Dating. Lewis carries a book called How to Meet Women. Lewis carries a book called Seduction in Three Easy Steps. Lewis carries a DVD called Sleepless in Seattle.

Gene is a man in Nancy Johnson Memorial Square. Gene carries a stamped envelope. Gene carries a DVD called Casablanca. Gene carries a coupon.

Rhoda is a woman in Marciony Street. Rhoda carries a book called The Marciony Street Murders. Rhoda carries a DVD called Unsolved Serial Killings XVIII. Rhoda carries a stamped envelope called a squashy package.

Martin is a man in Main Street. Martin carries a DVD called The Lifecycle of the South Sea Tortoise. Martin carries a coupon.

Antony is a man in Movie Rental. Antony carries a coupon. Antony carries a stamped envelope called a postcard.

Shelby is a man in the Town Hall. Shelby carries a DVD called Conducting An Orderly Meeting. Shelby carries 5 stamped envelopes. Shelby carries an ice cream cone. Shelby carries a coupon.

Christopher is a man in the Library. Christopher owns the Library.

Linnea is a woman in the Alfred Cralle Pool Hall. Linnea owns the Alfred Cralle Pool Hall.

Ned is a man in the Movie Rental Store. Ned owns the Movie Rental.

After printing the name of someone (called target) while listing contents: if the target owns the location of the target, say " (the owner)".

The description of a person is usually "[The noun] [if the noun is clean]looks healthy[otherwise]is the color of [infection color of the noun][end if]."

After examining another person who is carrying something: say "[if the noun is female]She[otherwise]He[end if] is carrying [a list of things carried by the noun]."

When play begins: let Patient Zero be a random other person; now patient zero is infected.

This is a light variation of a previous example, but we use it here because it is convenient:

Section 5 - Conversation

A person has a table name called conversation.

Instead of asking someone about something:
    let the source be the conversation of the noun;
    if topic understood is a topic listed in source:
        if there is a turn stamp entry:
            say "You have already heard that [summary entry].";
        otherwise:
            now turn stamp entry is the turn count;
            now the character entry is the noun;
            say "[reply entry][paragraph break]";
    otherwise:
        say "[The noun] stares at you blankly.".

Instead of telling someone about something:
    try asking the noun about it.

Understand "recap" or "recall" or "review" as recalling conversations.

Recalling conversations is an action applying to nothing.

Carry out recalling conversations:
    repeat with speaker running through other people:
        let source be the conversation of the speaker;
        sort source in turn stamp order;
        say "[The speaker] has so far told you: [line break]";
        let index be 0;
        repeat through source:
            if there is a turn stamp entry and the speaker is character entry:
                let index be 1;

say " [summary entry][line break]";
if index is 0, say " absolutely nothing[line break]";
say line break.

The conversation of a person is usually Table of General ChitChat.

Table of General ChitChat

[illegible]	[illegible]
[illegible]	"'Can you believe they think we didn't pay our taxes.'"
"'Got the highline tonight, only got a [illegible] power, har, har.'"	
"'Hear it's not expected to rain for 132 days,' replies [the noun]."	
"'Now the [illegible] is covered with [illegible] only."	
"'Heard a song [illegible] on the radio. Terrible,' [the noun] replies finally."	
"'Got the best ice cream in town,' says [the noun]. 'Best ice cream in town, and that's saying something, you bet.'"	
"'Vanilla with [illegible],' adds [the noun]. 'Not to everyone's taste, like...' [the noun as pronoun] considers for a moment. 'Like ginger ice cream. Big pieces of crystallized ginger... not everyone likes that.'"	
"'Oh, the weather outside is [illegible] since the gummint started making it for us,' says [the noun]. 'Now he just reads off the schedule on the air every morning. Pretty much takes the fun right out of the news, if you ask me.'"	
"'Have to [illegible] than what you get.'"	"'You'll be paying for something nicer don't you?'"
"'[illegible] replies [the noun][otherwise]Work at the creamery, like most folk around here,' answers [the noun]."	
"'[illegible] cookbook section," says [the noun]. 'That got censored way back when-- well, way back.'"	

The conversation of Vanessa is the Table of Vanessa Chatter.

Table of Vanessa Chatter

```
to stay
"[The
a person
ness"
```

```
to stay
"[The
a person
ness"
```

```
After reading a command:
    while player's command includes "the":
        cut the matched text.
```

This strips 'the' out of the command, so that ASK PERSON ABOUT THE RAIN will be understood as well as ASK PERSON ABOUT RAIN.

Now we try something a bit unusual. Inform on its own will report each action on its own line, so that each character who walks into or out of a room will be described in a separate paragraph. This is usually fine, but in a game with a lot of characters moving around simultaneously, it can become a bit overwhelming. Instead, we may want to condense these reports into a single line, such as "Ben and Jerry enter from the south". The following accomplishes that goal by replacing some of the reporting rules, storing the information in a table, and then reading the table back later, once all the character movement has been resolved and the reports can usefully be collated:

```
Section 6 - Movement Description
```

```
A person has some text called walk style. The walk style
of a man is usually "stride". The walk style of a woman
is usually "strut". The walk style of Gene is "[one
of]wander[or]stroll[purely at random]". The walk style of
Francine is "waddle". The walk style of Antony is
"scamper". The walk style of Rhoda is "sashay".
```

```
Table of Visible Exits

to
a
with 10 blank rows.
```

```
Table of Visible Entrances

[illegible] chosen
[illegible] a person
with 10 blank rows.



To clear (current table - a table name):
    repeat through current table:
        blank out the whole row.



To tidy departures of (current table - a table name):
    let next direction be up;
    repeat through current table:
        if heading chosen entry is next direction:
            let accomplice be character entry;
            choose row with heading chosen of next
direction in the current table;
            if total entry is 1:
                now second entry is accomplice;
                now total entry is 2;
            if total entry is 2:
                unless the second entry is accomplice:
                    now third entry is accomplice;
                    now total entry is 3;
            choose row with character of accomplice in the
current table;
            blank out the whole row;
        otherwise:
            let next direction be heading chosen entry.



A door has a person called last opener.



Report someone opening a door:
    now group size is 1;
    now the last opener of the noun is the person asked;
    if the person asked is visible, say "[The person asked]
opens [the noun]. [run paragraph on]" instead;
    otherwise say "[The noun] opens from the other side.
[run paragraph on]" instead.
```

```
Report someone going through a door (called route):
    if the person asked is not the last opener of the route,
continue the action;
    if the person asked is the last person named, say
"[The person asked as pronoun]";
    otherwise say "[The person asked]";
    say " [if the person asked is in the
location]comes[otherwise]goes[end if] through[if the last
thing named is not the route] [the route][end if]." instead.


The last thing named is a thing that varies. Before
printing the name of something (called target) which is
not a person: now the last thing named is the target.


Report someone going a direction:
    if the person asked is in the location,
        choose a blank row in the table of visible
entrances;
    otherwise choose a blank row in the table of visible
exits;
    now character entry is the person asked;
    now total entry is 1;
    if the person asked is in the location,
        now heading chosen entry is the opposite of the
noun;
    otherwise now heading chosen entry is the noun;
    stop the action.


This is the movement reporting rule:
    sort the Table of Visible Entrances in heading chosen
order;
    tidy departures of the table of visible entrances;
    sort the Table of Visible exits in heading chosen
order;
    tidy departures of the table of visible exits;
    let total row count be the number of filled rows in the
Table of Visible Entrances plus the number of filled rows
in the Table of Visible Exits;
    if total row count is 0, rule succeeds;
    generate descriptions from the Table of Visible
Entrances;
```

```
    generate descriptions from the Table of Visible Exits;
    clear the Table of Visible Entrances; clear the Table
of Visible Exits.



To generate descriptions from (current table - a table
name):
    let count be the number of filled rows in the current
table;
    if count is 0, rule succeeds;
    let index be count;
    repeat through the current table:
        let accomplice be character entry;
        if character entry is a person, now character entry
is marked for listing;
        if there is a second entry and second entry is a
person, now second entry is marked for listing;
        if there is a third entry and third entry is a person,
now third entry is marked for listing;
        let target be the room the heading chosen entry
from the location;
        if total entry is 3, say "[The character entry], [the
second entry][optional comma] and [the third entry] ";
        if total entry is 2, say "[The character entry] and
[the second entry] ";
        if total entry is 1:
            if the character entry is the last person named,
say "[The character entry as pronoun] ";
            otherwise say "[The character entry] ";
        if total entry is 1, say "[walk style of the character
entry]s ";
        otherwise say "walk[if total entry is 1]s[end if] ";
        if the character entry is in the location:
            if location is indoors and target is indoors, say
"over from ";
            if location is outdoors and target is indoors, say
"out of ";
            if location is indoors and target is outdoors, say
"in from ";
            if location is outdoors and target is outdoors, say
"over from ";
        otherwise:
            if location is indoors and target is indoors, say
"over to ";
            if location is outdoors and target is indoors, say
"into ";
            if location is indoors and target is outdoors, say
"out [if a door is visible][the random visible door][end if]
to ";
            if location is outdoors and target is outdoors, say
```

```
"over to ";
        if target is outdoors, say "[the heading chosen
entry]";
        otherwise say "[the target]";
        if the total entry is 1 and count is 1 and accomplice
carries something, say ", carrying [a list of things carried
by the accomplice]";
        decrement index;
        make delimiter index of count, continuing;
        now group size is total entry;
    if a marked for listing person is infected:
        [eliminate the case in which we have already seen
this description because we just typed LOOK and the
patient was in the room at the time]
        if looking and a marked for listing person is not in
the location:
            clear marked people;
            say paragraph break;
        otherwise:
            describe patients;
    otherwise:
        clear marked people;
        say paragraph break.



The last person named is a person that varies. Before
printing the name of a person (called target): now the
last person named is the target. Group size is a number
that varies. Group size is 1.



To clear marked people:
    repeat with named party running through people:
        now the named party is not marked for listing.



Before listing nondescript items:
    if the number of people who are marked for listing is
0, make no decision;
    say "You can see [a list of people who are marked for
listing] here. ";
    now group size is the number of people who are
marked for listing;
    describe patients;
    now every marked for listing person is not marked for
listing.
```

```
To describe patients:
    if every marked for listing person is infected and at
least three people are marked for listing:
        say "They are all sick as dogs, every one.";
        clear marked people;
        rule succeeds;
    otherwise:
        if the number of people who are marked for listing
is greater than two and the number of infected people
who are marked for listing is greater than the number of
clean people who are marked for listing:
            say "Only [the list of clean people who are
marked for listing] currently remain[if the number of
clean people who are marked for listing is 1]s[end if]
untainted.";
            clear marked people;
            rule succeeds;
    let count be the number of marked for listing other
people who are infected;
    if count is 0:
        say paragraph break;
        make no decision;
    let index be count;
    repeat with patient running through marked for listing
other people who are infected:
        if index is count:
            if count is 1 and the patient is the last person
named:
                say "[The patient as pronoun]";
            otherwise:
                say "[The patient]";
        otherwise:
            say "[the patient]";
        say " [looks as though dipped in for index]
[infection color of the patient]";
        decrement index;
        make delimiter index of count;
    clear marked people.



To say (named character - a man) as pronoun:
    if group size is 1, say "He"; if group size is 2, say "The
latter"; if group size is greater than 2, say "The last".



To say (named character - a woman) as pronoun: if
```

```
group size is 1, say "She"; if group size is 2, say "The
latter"; if group size is greater than 2, say "The last".



To say looks as though dipped in for (index - a number):
    let divider be the number of filled rows in the Table of
Dipping Phrases;
    if index is greater than 4, let index be the remainder
after dividing index by divider;
    choose row index in the Table of Dipping Phrases;
    say dipping entry.



Table of Dipping Phrases
dipping
"looks as though dipped in"
"could have been rolling in"
"has a bad case of"
"suffers from"
"contracted a virulent"



A door is usually scenery.
```

The next part could be simpler, but for rigor we will write it in such a way that it will work whether or not the serial comma is in use. This requires some extra work.

```
To make delimiter (index - a number) of (count - a
number), continuing or halting:
    if index is 0:
        if continuing, say ". [run paragraph on]";
        otherwise say ".";
    otherwise if index is 1:
        if count is 2, say " and ";
        otherwise say "[optional comma] and ";
    otherwise:
        say ", ".



To say optional comma:
    if the serial comma option is active:
        say ",".
```

Test me with "go to cold comfort / z / z / z / z / ask vanessa for french vanilla / ask vanessa for chocolate / ask vanessa about flavors / ask vanessa for chocolate chocolate chip".

Because so much of this game is randomized, it will not be possible to provide a test command that systematically solves it. A good strategy is to go to Main Street, get the syringe; go to the Town Hall and get the key; then visit the shops, inject everyone, and lock them in when they've all been injected. Then go to the Pool Hall, turn on the air conditioner, and wait for the remaining parties to show up.

This is also something that could get fairly slow if we added many more rooms and characters to it. In that case, we might want to select fast route-finding so that character movement won't take as long. This will cost memory, possibly forcing the game into Glulx format if it isn't already, but significantly reduce the run-time for large maps with numerous people moving each turn:

Use fast route-finding.

ExampleVirtue *WI*

Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

"Virtue"

The Cloister is a room. Lady Teresa is a woman in the Cloister. Mother Margaret is a woman in the Cloister.

Attacking someone is misbehavior. Kissing someone is misbehavior.

Instead of asking someone to try misbehavior: say "[The person asked] stiffens, offended by your suggestion."

Test me with "kiss margaret / margaret, kiss me".

Example **The Hypnotist of Blois**

A hypnotist who can make people obedient and then set them free again.

"The Hypnotist of Blois"

A person is either hypnotized or alert. A person is usually alert.

Persuasion rule for asking a hypnotized person to try doing something:
    persuasion succeeds.

Understand "hypnotize [someone]" as hypnotizing.

Hypnotizing is an action applying to one thing.

Check hypnotizing:
    if the noun is hypnotized, say "Already done." instead.

```
Carry out hypnotizing:
    now the noun is hypnotized.


Report hypnotizing:
    say "[The noun] slips under your control."


Instead of waking someone hypnotized:
    now the noun is alert;
    say "[The noun] returns abruptly to consciousness."
```

We will learn more about check rules for other characters shortly, but the following prevents the hypnosis patient from trying to hypnotize us in turn:

```
Check someone hypnotizing someone:
    stop the action.


Maison de la Magie is a room. "In a darkened room, a
few hundreds of paces from the chateau of Blois, you
give to tourists three shows a day: displaying to them
power they do not comprehend and spectacles they do
not deserve."


A volunteer is a woman in the Maison. "A volunteer from
the audience stands facing you, [if alert]skeptically
awaiting hypnosis[otherwise]her face worshipful and
obedient[end if]." The printed name of the volunteer is
"volunteer from the audience". The description is "A
distracted, susceptible woman." The volunteer wears a
t-shirt and a baseball hat.


The player wears a top hat and a cape.


Test me with "volunteer, remove hat / hypnotize
```

volunteer / look / volunteer, remove hat / wake volunteer / volunteer, wear hat".

Example: Latin Lessons

Supplying missing nouns and second nouns for other characters besides the player.

WI

If we're defining actions for other characters to follow, we may want to include them in our "rule for supplying a missing noun". We can do this if we write our "while..." clause to apply to any actor, as follows:

```
"Latin Lessons"

The Latin Studio is a room. Rick is a man in the Studio.

A dance-name is a kind of thing. Argentine tango, samba, merengue, cha-cha, street salsa are dance-names.

Dancing is an action applying to one visible thing.
Understand "dance [any dance-name]" as dancing.
Understand "dance" as dancing.

Rule for supplying a missing noun while an actor dancing:
    now the noun is street salsa.

Report someone dancing:
    say "[The actor] dances a few steps of [the noun] for you."
```

Report dancing:
    say "You dance a few steps of [the noun]."

Persuasion rule for asking someone to try dancing:
persuasion succeeds.

Test me with "dance / dance samba / rick, dance / rick, dance merengue".

 Example **Generation X** *WI*

A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

"Generation X"

The Volunteer Center is a room. "A fairly spartan office, though there are a few attractive posters from advertising campaigns of the past, and an ominous map charting the deforestation of Brazil."

The desk is scenery in the Volunteer Center. "Your standard metal desk, with a drawer for postal supplies and the like." On the desk is a pile of leaflets and a pile of business cards. The power stapler is a device on the desk. A drawer is part of the desk. It is openable and closed. In the drawer are a sponge and a roll of bulk-mail stamps.

Instead of doing something:

```
    if examining, continue the action;
    if searching, continue the action;
    if looking, continue the action;
    if asking Jenna to try doing something, continue the
action;
    say "The whole point of this exercise is to train Jenna.
Once you have her on the envelope-stuffing, you can go
make some calls."



Instead of answering Jenna that something:
    say "You're going to have to break the instructions
down to the simplest ones possible, given that she's in
the mood to misunderstand on purpose."



A cardboard box is in the Center. In the cardboard box
is a pile of empty envelopes. The box is openable and
closed. The description is "A recycling symbol is
prominent on the side, which makes you feel a little bit
better about using a mailing campaign for this cause."



Jenna is a woman in the Volunteer Center. "Your
daughter Jenna is here, barely visible through the cloud
of resentment." The description is "Jenna [if jenna
carries something]has [the list of things carried by
Jenna][otherwise]returns your stare, with added
hostility[end if]."



A persuasion rule for asking Jenna to try doing
something: persuasion succeeds.



Unsuccessful attempt by Jenna doing something:
    repeat through Table of Retorts:
        if the reason the action failed is the cause entry:
            say "[response entry][paragraph break]";
            rule succeeds;
    say "'I don't see how I'm supposed to do that, Mom,'
Jenna says."
```

Table of Retorts

response
"'Is that like, a metaphor or something?' Jenna asks, momentarily diverted."
"[any kind of impossibility]"
"[any kind of impossibility]"
"[already worn]"
"[already worn]"
"[already dropped]"
"[already dropped]"
"[already worn]"
"[already worn]"
"[not worn]"
"[not worn]"
"[already closed]"
"[already closed]"
"[already open]"
"[already open]"
"[already off]"
"[already off]"
"[already on]"
"[already on]"
"[already unlocked]"
"[already unlocked]"
"[already locked]"
"[already locked]"

To say physical impossibility:
    say "'Maybe you should've brought someone a little stronger,' Jenna says. 'Like the Incredible Hulk.' "

To say already done:
    repeat through Table of Bored Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "'Okay, I'm going to be, like, in the car,' says Jenna. 'Outside.' ";
    end the story saying "Jenna has gotten fed up".

Table of Bored Remarks
response
"'Did that,' says Jenna."
"'Check,' says Jenna."
"'Yeah, Mom, I already did that, okay?'"
"'Look, if I have to be here doing dumb stuff, could you at least tell me to do stuff that isn't already done?' Jenna asks wearily."
"Jenna gives a great upward sigh, riffling her bangs. 'Once again... there is totally no point.'"

Test me with "e / x jenna / jenna, get stapler / jenna, get stapler / x jenna / jenna, drop stapler / jenna, drop stapler / jenna, open box / jenna, open box / jenna, switch stapler on / jenna, switch stapler on / jenna, take desk / jenna, open box / jenna, open box".

ExampleNorthstar

Making Inform understand ASK JOSH TO TAKE INVENTORY as JOSH, TAKE INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

Most of the time, Inform understands commands to other characters when they take the form "JOSH, TAKE INVENTORY" or "JOAN, WEAR THE ARMOR". But novice players might also try commands of the form ASK JOSH TO TAKE INVENTORY or ORDER JOAN TO WEAR THE ARMOR.

The easiest way to make Inform understand such commands is to meddle directly with the player's command, changing it into the format that the game will understand, as here:

"Northstar"

The Northstar Cafe is a room. "The Northstar is crammed with its usual brunch crowd, and you were lucky to get a table at all. You are now awaiting the arrival of your ricotta pancakes."

Josh is a man in The Northstar Cafe. "Josh is on his way past your table." The description of Josh is "He is a waiter here, but you also know him socially, so he tends to be more chatty than the other waiters." A persuasion rule: persuasion succeeds.

```
After reading a command:
    let N be "[the player's command]";
    replace the regular expression "\b(ask|tell|order) (.+?)
to (.+)" in N with "\2, \3";
    change the text of the player's command to N.



Test me with "ask Josh to take inventory / tell Josh to
take inventory / order Josh to take inventory".
```

Note that we have to copy N back explicitly to replace the player's command.

ExampleFor Demonstration Purposes

A character who learns new actions by watching the player performing them.

Suppose we want to have a character who can dynamically learn new actions by observing the player performing them. We could do this by adding the actions to a list of things the character can do, but using a relation to express the same idea allows for tidier, easier-to-read code.

Thanks to Jesse McGrew for the initial design of this example.

```
"For Demonstration Purposes"



Section 1 - Procedure



Capability relates various people to various stored
actions. The verb to be capable of means the capability
relation.



Persuasion rule:
    let CA be the current action with no specific actor;
    if the person asked is capable of CA:
```

```
        persuasion succeeds;
    otherwise:
        say "[The person asked] look[s] confused. Maybe a
demonstration would help.";
        persuasion fails.
```

The action requester is an object that varies. The action requester variable translates into I6 as "act_requester".

```
To decide which action is the current action with no
specific actor:
    let old actor be the person asked;
    let old requester be the action requester;
    now the person asked is the player;
    now the action requester is nothing;
    let CA be the current action;
    now the person asked is the old actor;
    now the action requester is the old requester;
    decide on CA.
```

The learning by observation rule is listed after the report stage rule in the specific action-processing rules.

Definition: a person is other if he is not the player.

```
This is the learning by observation rule:
    repeat with the viewer running through other people
who can see the player:
        if the player is the actor and viewer is not capable
of the current action:
            say "[The viewer] watches your behavior with
interest. Seems like [they] [are] learning.";
            now the viewer is capable of the current action.
```

Section 2 - Scenario

The Daily Planet is a room. Clark is here. He is a man.

When play begins:
    now Clark is capable of taking inventory.

Test me with "Clark, inventory / Clark, x me / x me / Clark, x me".

Example **Under Contract** *WI*

Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

"Under Contract"

The Sound Stage is a room. The description is "A somewhat creaky and unconvincing reproduction of the docks of San Francisco."

Clark Gable is a man in the Sound Stage. "Clark is here, carrying [the list of things carried by Clark][if Clark wears anything] and wearing [the list of things worn by Clark] [end if]." Clark carries a swagger stick. Clark carries an open openable player's holdall called a briefcase. The briefcase contains a signed contract. Clark wears a fedora and a pair of pants. A handle is part of the briefcase.

The player carries an open openable player's holdall called a frilly bag. The carrying capacity of the player is

2. The player wears a sparkly scarf and a slinky dress. A strap is part of the bag. The tent is a portable enterable container.

Instead of taking off the slinky dress: say "Rowr! Not that kind of movie, babe."

Instead of taking off the pants: say "That's not your job, whatever the society pages might suggest."

Unsuccessful attempt by Clark doing something:
    repeat through table of Clark Retorts:
        if the reason the action failed is the cause entry:
            say "[response entry][paragraph break]";
            rule succeeds;
    say "'I don't think that's in the script,' says Clark dubiously."

Table of Clark Retorts

cause	response
[an actor taking yourself]	"I'm all yours, darling," Clark remarks. You've heard that line before, but it sounds so much more convincing from him."
[an actor taking other people]	"I don't think you'd be appreciated if I tried to do that to [the noun]," he rumbles."
[an actor trying to take the pants]	"I don't want to pop [the pants] out," Clark remarks."
[an actor trying to take people]	"I don't take people across state lines, thief," Clark replies. 'It ain't proper.'"
[an actor taking something fixed in place]	"Do you see where I'm standing here?" Clark demands."
[an actor taking what's already taken]	"I'm already holding it."
[an actor taking scenery]	"I'm not the scenery male, darling,' he says with a wry twinkle."
[an actor taking what's fixed in place]	"I'm not the scenery male, darling,' he says with a wry twinkle."
[an actor trying to carry too many things]	Clark cries, "I've only got so many hands, darling," he says."
[an actor trying impossible containers]	"I'm sorry, that's impossible."
[an actor trying impossible]	"I'm sorry, that was impossible."
[an actor trying impossible doors]	"I'm sorry, that's impossible."
[an actor trying impossible containers]	"I'm sorry, that's impossible."
[an actor trying impossible others]	"I'm sorry, that's impossible."
[an actor dropping themself]	"We're all in this together, me and me,' Clark replies, with a smile."
[an actor dropping what's already dropped]	"I already have it."
[an actor dropping scenery]	"Ain't dropping what's mine," replies Clark."
[an actor dropping clothes being worn]	"I'm not undressing in front of the audience."

"[a] humble gesture, says Clark. Oh, he's so modest."
"[The second] won't support the thing,' says Clark reprovingly."
[an] ridiculous [request] being worn rule
[an] indecent [clothing] is being worn rule
[an] obscene [item] is being worn rule
"Or wearing just [some] clothing every year,' says Clark. 'In short: no.'"
[already done] what's already worn rule
"What, you want me to [eat] poison me?'"
[an] edible [thing] without removing it first rule
[already done] what's not worn rule
[already done] what's already closed rule
[already done] what's already open rule
[already done] what's already off rule
[already done] what's already on rule
[already done] what's already unlocked rule
[already done] what's already locked rule

To say already done:
    repeat through Table of Clark's Bored Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "'Already done.' "

Table of Clark's Bored Remarks
response
"'I have anticipated your every demand.'"
"'We've been through this part of the script already,' Clark remarks, with just a hint of creeping tedium."
"'Right right, already taken care of...'"
"'I'm a believer in rehearsal,' Clark says, sounding more bored than ever, 'but we really have got all this done already.'"

To say salacious retort:
    repeat through Table of Clark's Flirtatious Remarks:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "Clark, driven past the point of endurance, seizes you in his arms and kisses you.";
    end the story finally.

Table of Clark's Flirtatious Remarks
response
"Clark allows his mustache to quirk at the suggestion."
"Clark wiggles his eyebrows at you."
"'That kind of thing isn't in my contract, sweetie,' says Clark. 'If you're interested you'll have to ask off-camera.'"
"'I'd feel so... bare,' Clark says, with a sidelong look at you."
"'You first,' says Clark."
"Clark's stare is intense. You may be on the verge of breaking him."

To say physical impossibility:
    repeat through Table of Clark's Frustrated Denials:
        say "[response entry]";
        blank out the whole row;
        rule succeeds;
    say "Clark makes a helpless gesture about his ability to move himself, or parts of himself, through solid objects. "

Table of Clark's Frustrated Denials
response
"Clark frowns. 'I can't reach through things -- unless you were planning a special effect?'"
"'Look,' says Clark. 'If you want a stage magician, hire one.'"
"Clark just groans."
"Clark says drily, 'Is this in the script, or are we doing improvisation now?'"

The block giving rule is not listed in the check giving it to rules. The block showing rule is not listed in the check showing it to rules. The report smelling rule is not listed in the report smelling rules. The report listening rule is not listed in the report listening to rules.

Carry out listening to something:

```
    do nothing.

Report listening to something: say "Your attention bears
no interesting result."

Carry out smelling something:
    do nothing.

Report smelling something: say "Your attention bears
no interesting result."

Report someone listening to something: say "[The
person asked] concentrates, listening."

Report someone smelling something: say "[The person
asked] sniffs at [the noun]."

Instead of asking someone for something: try asking the
noun to try giving the second noun to the player.

Carry out showing something to someone: say "You
reveal [the noun] to [the second noun]."

Carry out Clark showing something to someone:
    if the second noun is the player:
        say "Clark shows you [the noun]. [run paragraph
on]";
        try examining the noun;
    otherwise:
        say "Clark reveals [the noun] to [the second noun]."
```

```
Instead of asking someone to try saying yes: try saying
yes. Instead of asking someone to try saying no: try
saying no. Instead of asking someone to try saying
sorry, try saying sorry.
```

And now we make Clark a very amenable type:

```
A persuasion rule for asking people to try doing
something: persuasion succeeds.
```

There are quite a few actions for which no automatic other-character behavior is provided at all, because they always end in failure when the player tries them. Currently the response is not very interesting. Let's spice it up a bit:

```
Burning something is useless action. Waking up is
useless action. Thinking is useless action. Cutting is
useless action. Jumping is useless action. Tying
something to something is useless action. Drinking
something is useless action. Swinging is useless action.
Rubbing is useless action. Setting something to
something is useless action. Waving hands is useless
action. Buying is useless action. Climbing is useless
action. Sleeping is useless action. Kissing is useless
action. Throwing something at something is useless
action. Attacking is useless action. Asking something
about something is useless action. Telling something
about something is useless action. Answering
something that something is useless action. Waking
something is useless action.
```

```
A persuasion rule for asking people to try useless
action:
    say "Clark Gable frowns. 'I don't mean to be difficult,
but I can't see any point in that.'";
    persuasion fails.
```

```
A persuasion rule for asking Clark to try taking off the
pants:
```

```
    say "He gives you a look of mocking exasperation.
Apparently you'll have to take a (slightly) more subtle
approach.";
    persuasion fails.
```

```
    Check someone giving something to someone (this is
the can't give worn items rule):
        if the person asked wears the noun, stop the
action.
```

And because we do not want Clark automatically taking off the pants as the result of a drop action:

```
Instead of Clark taking off the pants: do nothing.
```

We know that this can only come about as a default action, because we have arranged matters so that he cannot be persuaded directly.

```
Test me with "give dress to clark / clark, take off pants /
clark, eat pants / clark, give me the pants / clark, drop
pants / clark, put the pants in the briefcase / again / g / g
/ g"
```

Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

We have already seen how Before... rules can generate implicit actions for the player, so that the player will, for instance, open doors before trying to walk through them. The same can be done for characters other than the player; so for instance:

```
"IQ Test"
```

The Donut Shop is a room. "Vibrantly decorated in donut colors: pink, brown, and cream."

Ogg is a man in the Donut Shop. "Ogg is slumped in the corner[if Ogg carries something] with [a list of things carried by Ogg][end if]. He wears a nametag which says 'HELLO MY NAME IS OG.'" Understand "og" as Ogg. Ogg wears a nametag. The description of the nametag is "Sadly misspelled."

The Donut Shop contains a transparent closed openable locked lockable container called a case. The case contains some cake donuts. The donuts are edible.

The matching key of the case is a silver key. The silver key is carried by Ogg.

A persuasion rule for asking someone to try doing something:
    persuasion succeeds.

Before someone opening a locked thing (called the sealed chest):
    if the person asked is carrying the matching key of the sealed chest, try the person asked unlocking the sealed chest with the matching key of the sealed chest;
    if the sealed chest is locked, stop the action.

Before someone taking something which is in a closed container (called the shut chest):
    try the person asked opening the shut chest;
    if the shut chest is closed, stop the action.

```
The block giving rule is not listed in the check giving it to
rules.


Test me with "open case / get donuts / og, get donuts /
og, give me the donuts / eat donuts".
```

(A more detailed set of before rules for people using doors and locks is provided in Emily Short's Locksmith extension.)

---

203 **Example** **The Man of Steel** *WI*

An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

It will sometimes be handy to write actions that are only available to the non-player characters and not to the player. To do this, we just define an action which has no "understand": the player will never be able to specify this command himself, but other characters can try it.

This is particularly useful for creating abstract or general actions for when we want a character to eat something, wear something, or go somewhere, but aren't too particular as to what the object is; as here, where we just want Clark to move away from the kryptonite, regardless of direction:

```
"The Man of Steel"


Escaping is an action applying to nothing.


Carry out someone escaping:
    let space be the holder of the person asked;
    let place be a random room which is adjacent to the
space;
    let way be the best route from the space to the place;
    try the person asked going way.
```

```
Every turn:
    if Clark Kent can see kryptonite, try Clark Kent
escaping.

The Daily Planet Newsroom is a room.

Perry White's Office is west of the Newsroom. In Perry
White's Office are a desk and a poster of Elvis. On the
desk is a lead-lined box. The box is openable. In the box
is the green kryptonite crystal.

The Supply Closet is east of the Newsroom. The
Elevator Bank is north of the Newsroom.

Clark Kent is a man in the Newsroom. "Clark [if Clark
can see the kryptonite]looks a bit ill[otherwise]is here,
frowning as he revises his latest article[end if]."

Test me with "west / get box / east / close box / east /
west / north / south / west".
```

Example **The Man of Steel Excuses Himself**

Elaborating the report rules to be more interesting than "Clark goes west."

Report rules can be a good point at which to add local color: while Inform provides default descriptions of character behavior, these are rather generic and can stand to be customized.

For instance, if we wanted to liven up our previous Clark Kent example:

```
Report Clark Kent going a direction (called the way):
```

```
    say "[one of]With a particularly weak
excuse[or]Muttering[at random] about [random excuse
subject], Clark heads [way]." instead.


To say random excuse subject:
    choose a random row in the Table of Lame Excuses;
say "[reply entry]".


Table of Lame Excuses
reply
"needing a paper-clip"
"wanting an English-Tuvalu dictionary"
"walking a neighbor's dog"
"hearing air-raid sirens"
"having drunk too much coffee"
"thinking he smells smoke"
"wondering where Lois got to"
"needing to speak to Jimmy"
"noticing the Good Year blimp"
```

It's good to be careful, as the library report rules have been designed and tested to describe every contingency (going through doors, going in vehicles, etc.): so when replacing a report rule, we should try to consider all the possible variations of the action that we might want to describe.

However, in this case, our scenario is so simple that there are no doors, vehicles, or pushable objects, so we're safe in giving Clark a very simple reporting scheme.

Example **Boston Cream**

A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

```
"Boston Cream"


Use scoring.
```

The Donut Shop is a room. "Vibrantly decorated in donut colors: pink, brown, and cream."

Ogg is a man in the Donut Shop. "Ogg is slumped in the corner[if Ogg carries something] with [a list of things carried by Ogg][end if]. He wears a nametag which says 'HELLO MY NAME IS OG.'" Understand "og" as Ogg. Ogg wears a nametag. The description of the nametag is "Very neatly written."

The Donut Shop contains a transparent closed openable locked lockable enterable container called a case. The case is fixed in place. The case contains some cake donuts, some jelly donuts, and some apple fritters. "The [if unopenable]damaged[otherwise]gleaming[end if] donut case [if something is in the case]contains [a list of things in the case][otherwise]has been stripped of its contents[end if]." The cake donuts, the jelly donuts, and the apple fritters are edible.

The matching key of the case is a silver key. The silver key is in a mesh basket. The mesh basket is closed, transparent, and openable. It is in the Donut Shop.

Before someone taking something which is carried by the player:
    if the person asked cannot touch the player:
        say "Ogg looks with a fixed frown at [the noun].";
        continue the action;
    say "[The person asked] comes up and shakes your arm until you drop [the noun].";
    say line break;
    silently try dropping the noun;
    stop the action.

Before someone unlocking a locked thing with

```
something which is not carried by the person asked:
    try the person asked taking the second noun;
    stop the action.



Before someone opening a locked thing (called the
sealed chest):
    if the person asked can see the matching key of the
sealed chest:
        if the matching key of the sealed chest is enclosed
by the sealed chest:
            say "Seeing the paradoxical location of [the
matching key of the sealed chest], [the person asked]
gives a howl of rage.";
            increment the score;
            end the story finally saying "You have thwarted
Ogg";
        otherwise:
            try the person asked unlocking the sealed chest
with the matching key of the sealed chest;
        stop the action.



Before someone taking something which is in a closed
container (called the shut chest):
    try the person asked opening the shut chest;
    stop the action.



Ogg has a number called hunger. The hunger of Ogg is
0.



Every turn:
    increment the hunger of Ogg;
    if the hunger of Ogg is 2 and Ogg is visible, say
"Ogg's stomach growls.";
    if the hunger of Ogg is 3 and Ogg is visible:
        if Ogg can see an edible thing (called the target):
            say "Ogg eyes [the target] with obvious intent.";
        otherwise:
            say "Ogg glances at you in a disturbingly shifty
way.";
    if the hunger of Ogg is greater than 3:
        if Ogg carries an edible thing (called the target):
```

```
            try Ogg eating the target;
        otherwise:
            let new target be a random edible thing which
can be seen by Ogg;
            if the new target is a thing:
                try Ogg taking the new target;
            otherwise:
                if Ogg can touch the player, end the story
saying "Ogg is gnawing your ankle";
                otherwise try Ogg taking the player.
```

The crumbs are a thing. "Crumbs of [the list of edible things which cannot be seen by the player] lie scattered over the whole floor."

Instead of asking Ogg to try doing something when Ogg cannot touch the player:
    say "Ogg tilts his head and shrugs, unable to hear your instruction clearly."

Instead of asking Ogg to try eating something:
    say "It's not as though Ogg really needs any encouragement in that department, is it?"

Definition: Ogg is hungry if the hunger of Ogg is greater than 2.

Persuasion rule for asking Ogg to try doing something: if Ogg is hungry, persuasion fails; persuasion succeeds.

Persuasion rule for asking Ogg to try giving something edible to the player:
    persuasion fails.

Unsuccessful attempt by Ogg doing something:
    if the reason the action failed is a failing listed in the Table of Ogg Retorts:
        say "[reply entry][paragraph break]";
    otherwise:
        say "Ogg looks adorably confused."

Table of Ogg Retorts

failing	reply
can't take people's possessions rule	"Ogg take people."
can't take what's already held rule	"Ogg take that already."
can't take scenery rule	"That's scenery, you heavy.'"
can't take what's fixed in place rule	"That's fixed in place."
can't drop what's not held rule	"Huh?"

Carry out Ogg eating an edible thing:
    move the crumbs to the holder of Ogg;
    now the hunger of Ogg is 0.

Report Ogg unlocking something with something:
    say "Ogg struggles a bit with [the second noun] in the lock of [the noun], but does eventually succeed. 'Hunh!' says Ogg.";
    stop the action.

Carry out Ogg opening the case when the case has been open:
    now the case is unopenable.

Instead of closing the unopenable case:
    say "The glass panels are no longer properly seated in their tracks, and the case cannot be closed ever again."

Report Ogg opening the unopenable case:
    say "Ogg forces [the case] so hard that it does break.";
    stop the action.

Report Ogg opening the case:
    say "Ogg yanks [the noun] open with such force that you fear for its structural integrity.";
    stop the action.

Report Ogg taking something edible:
    say "Ogg acquires [the noun] with a look of tender affection.";
    stop the action.

Report Ogg eating something:
    say "Ogg chows down on [the noun], scattering crumbs in all directions.";
    stop the action.

Report Ogg eating something when the number of visible edible things is 1:
    say "Ogg eats [the noun] in his trademark style. You can no longer bear to watch.";
    stop the action.

After entering the case:
    say "You climb inside the case, folding yourself up uncomfortably."

After locking the case with something when the player is in the case:
    say "You turn the key firmly in the lock -- amazing it locks from within, but it does -- and settle yourself for a long wait, hoping this thing is not air-tight."

The maximum score is 1.

Test me with "open mesh / get key / unlock case / open case / enter case / close case / lock case / wait / wait".

---

229 Example Unthinkable Alliances *WI*

People are to be grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give notice of quitting, and to become a lone wolf.

The following is best tested by experimentally kissing and/or attacking, and typing RELATIONS after every change to see the effect.

"Unthinkable Alliances"

Unthinkable Solutions is a room. Sophie, Daisy, Ryan and Owen are in Unthinkable Solutions.

Alliance relates people to each other in groups. The verb to help means the alliance relation.

Sophie helps Ryan. Daisy helps Ryan. Owen helps the player.

Instead of kissing someone (called the blessed one):
    say "Smack!";
    now the player helps the blessed one.

Instead of attacking someone (called the vilified one):
    say "Smack!";
    now the player does not help the vilified one.

Test me with "relations / kiss sophie / relations / hit ryan / relations".

Example **The Abolition of Love** *WI*

A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

Suppose we are modeling a complex society seething with interpersonal relations of every kind.

"The Abolition of Love"

Section 1 - Relation types

Loving relates one person to one person.

Noticing relates various people to one person.

Impressing relates one person to various people.

Fancying relates various people to various people.

Acquaintance relates people to each other.

Marriage relates one person to another.

Alliance relates people to each other in groups.

The Chapel is a room. Elizabeth, Wickham and Darcy are people in the Chapel. Mr Bennett and Mrs Bennett are people in the Chapel. Georgiana is a person in the Chapel.

The verb to love means the loving relation.

The verb to notice means the noticing relation.

The verb to impress means the impressing relation.

The verb to fancy means the fancying relation.

The verb to know means the acquaintance relation.

The verb to be married to means the marriage relation.

The verb to be related to means the alliance relation.

```
Elizabeth loves Darcy. Elizabeth fancies Darcy.
Elizabeth notices Darcy. Elizabeth impresses Darcy.
```

```
Mr Bennett is related to Mrs Bennett and Elizabeth. Mr
Bennett is married to Mrs Bennett.
```

```
Georgiana is related to Darcy.
```

Now we want ways to set and unset all of these relations. (In the interests of thoroughness, we may get a bit far-fetched here. It is not recommended in practice that we make the player guess the verb "traduce".)

```
Section 2 - Setting and Unsetting Love (1-1)
```

```
Understand "infatuate [someone] with [someone]" as
infatuating it with. Infatuating it with is an action applying
to two visible things.
```

```
Carry out infatuating it with:
    now the noun loves the second noun.
```

```
Report infatuating it with:
    say "Now [the noun] loves [a random person loved by
the noun][if the second noun loves someone], while [the
second noun] loves [a random person loved by the
second noun][end if]."
```

```
Understand "embitter [someone] toward [someone]" as
embittering it toward. Embittering it toward is an action
applying to two visible things.
```

```
Carry out embittering it toward:
    now the noun does not love the second noun.

Report embittering it toward:
    say "[The noun] sees [the second noun] in a different
light and no longer feels any affection."
```

Because love is a 1-1 relation, a person cannot love more than one other character at a time. Whenever we set a character to love a new person, that person ceases to love the character loved before. It is a fickle world.

One to various relations are a bit more open: we can say someone impresses multiple other characters, and our additions to the list do not override the initial ones.

```
Section 3 - Setting and Unsetting Impressed (1-V)

Understand "commend [someone] to [someone]" as
commending it to. Commending it to is an action
applying to two visible things.

Carry out commending it to:
    now the noun impresses the second noun.

Report commending it to:
    say "[The second noun] takes a very decided interest
in [the noun]."

Understand "traduce [someone] to [someone]" as
traducing it to. Traducing it to is an action applying to
two visible things.

Carry out traducing it to:
    now the noun does not impress the second noun.
```

```
Report traducing it to:
    say "[The second noun], hearing your story, decides
not to be at all impressed with [the noun]."
```

And because this is a one-to-various relation, we can also make statements which set multiple relations at once, so:

```
Understand "celebrate [someone]" as celebrating.
Celebrating is an action applying to one visible thing.


Carry out celebrating:
    now the noun impresses every person.


Report celebrating:
    say "[The list of people who are impressed by the
noun] take a very decided interest in [the noun]."


Understand "slander [someone]" as slandering to.
Slandering to is an action applying to one visible thing.


Carry out slandering to:
    now every person is not impressed by the noun.


Report slandering to:
    say "Now [the noun] impresses [the list of people who
are impressed by the noun]."
```

Note that the above unsetting is not equivalent to "now the noun does not impress every person" -- which would be ambiguous in spoken English, as well. Various-to-one relations are similar:

Understand "draw the attention of [someone] to [someone]" as drawing the attention of it to. Drawing the attention of it to is an action applying to two visible things.

Carry out drawing the attention of it to:
    now the noun notices the second noun.

Report drawing the attention of it to:
    say "[The noun] glances thoughtfully in the direction of [the second noun]."

Understand "distract [someone] from [someone]" as distracting it from. Distracting it from is an action applying to two visible things.

Carry out distracting it from:
    now the noun does not notice the second noun.

Report distracting it from:
    say "You distract [the noun] from [the second noun]."

Understand "draw attention to [someone]" as drawing attention to. Drawing attention to is an action applying to one visible thing.

Carry out drawing attention to:
    now every person notices the noun.

Report drawing attention to:
    say "You quickly cause everyone to attend to [the noun]."

Understand "outshine [someone]" as outshining. Outshining is an action applying to one visible thing.

Carry out outshining:
    now every person does not notice the noun.

Report outshining:
    say "You quickly distract everyone from [the noun]."

Section 5 - Setting and Unsetting Fancying (V-V)

Understand "flatter [someone]" as flattering. Flattering is an action applying to one thing.

Carry out flattering:
    now every person fancies the noun.

Report flattering:
    say "You draw down universal admiration for [the noun] by casting him or her in a flattering light."

Understand "unflatter [someone]" as unflattering. [Okay, okay, but it's four am.] Unflattering is an action applying to one thing.

Carry out unflattering:
    now every person does not fancy the noun.

Report unflattering:
    say "[The noun] gives everyone a universal disgust."

Understand "admire [someone]" as admiring. Admiring is an action applying to one thing.

Carry out admiring: now the player fancies the noun.

Report admiring: say "You find you rather fancy [the noun]."

Understand "loathe [someone]" as loathing. Loathing is an action applying to one thing.

Carry out loathing: now the player does not fancy the noun.

Report loathing: say "You take [the noun] in disgust."

Understand "cause chaos" as causing chaos. Causing chaos is an action applying to nothing.

Carry out causing chaos:
    now every person fancies every person.

```
Report causing chaos: say "Now everyone fancies
everyone else, which is quite an inconvenient state of
affairs."

Understand "relieve chaos" as relieving chaos.
Relieving chaos is an action applying to nothing.

Carry out relieving chaos:
    now every person is fancied by no one.

Report relieving chaos: say "Now no one fancies
anyone, which is safe but tedious."
```

Our options for setting and unsetting symmetrical relations are more limited again:

```
Section 6 - Setting and Unsetting Marriage (1-1
Symmetrical)

Understand "marry [someone] to [someone]" as uniting
it in matrimony with. Uniting it in matrimony with is an
action applying to two visible things.

Carry out uniting it in matrimony with:
    now the noun is married to the second noun.

Report uniting it in matrimony with:
    say "You perform the marriage of [the noun] to [the
second noun], joining them to the family of [a list of
people related to the noun]."

Understand "divorce [someone] from [someone]" as
```

```
divorcing it from. Divorcing it from is an action applying
to two visible things.


Check divorcing it from:
    if the noun is not married to the second noun, say
"[The noun] is not married to [the second noun]
anyway." instead.


Carry out divorcing it from:
    now the noun is not married to the second noun.


Report divorcing it from:
    say "[The noun] is now not married to [the second
noun]."
```

When we unset the symmetrical relation on one side, it is automatically set or unset on the other. It is not necessary to say "the second noun is married to the noun" or "the second noun is not married to the noun", even though that is the case.

```
Section 7 - Setting and Unsetting Acquaintance (V-V
Symmetrical)


Understand "introduce [someone] to [someone]" as
introducing it to. Introducing it to is an action applying to
two visible things.


Carry out introducing it to:
    now the noun knows the second noun.


Report introducing it to:
    say "You introduce [the noun] to [the second noun].
Now [the noun] is acquainted with [the list of people who
are known by the noun], and [the second noun] is
```

```
acquainted with [the list of people who are known by the
second noun]."


Understand "announce [someone]" as announcing.
Announcing is an action applying to one visible thing.


Carry out announcing:
    now every person knows the noun.


Report announcing:
    say "You announce [the noun] to the whole
assembled company."


Understand "ostracise [someone]" as ostracising.
Ostracising is an action applying to one visible thing.


Carry out ostracising:
    now every person does not know the noun.


Report ostracising:
    say "You cause everyone present to forget and
pretend not to be acquainted at all with [the noun]."
```

And finally, setting groups:

```
Section 8 - Setting and Unsetting Familial Relations
(Groups)


Understand "make [someone] adopt [someone]" as
forcing it to adopt. Forcing it to adopt is an action
applying to two visible things.
```

```
Carry out forcing it to adopt:
    now the noun is related to the second noun.


Report forcing it to adopt:
    say "Now [the second noun] is related to [the list of
people related to the second noun]."


Understand "make [someone] disown [someone]" as
forcing it to disown. Forcing it to disown is an action
applying to two visible things.


Carry out forcing it to disown:
    now the second noun is not related to the noun.


Report forcing it to disown:
    say "Now [the second noun] is related to [the list of
people who are related to the second noun], and [the
noun] is related to [the list of people who are related to
the noun]."
```

Notice that when we say "the second noun is not related", we remove that person from the group: they are now in a separate group of their own, while the rest of the group's members remain related to one another.

And finally, a long litany of test cases, complete with the relations lists:

```
Test acquaintance with "relations / introduce darcy to
elizabeth / introduce darcy to wickham / announce mr
bennett / relations / ostracise wickham / introduce
georgiana to wickham / relations".


Test impression with "commend georgiana to elizabeth /
relations / celebrate Mrs bennett / relations / traduce
```

mrs bennett to darcy / relations / slander mrs bennett / relations".

Test notice with "draw the attention of darcy to elizabeth / relations / draw attention to mr bennett / relations / distract darcy from mr bennett / relations / outshine mr bennett / relations".

Test love with "embitter elizabeth toward darcy / relations / infatuate elizabeth with wickham / relations".

Test marriage with "marry elizabeth to darcy / relations / divorce elizabeth from darcy / relations".

Test alliance with "make mr bennett adopt georgiana / relations / make mrs bennett disown georgiana / relations".

Test fancying with "admire elizabeth / relations / loathe elizabeth / relations / flatter elizabeth / relations / unflatter elizabeth / relations / cause chaos / relations / relieve chaos / relations".

Test me with "test acquaintance / test impression / test notice / test love / test alliance / test fancying / test marriage".

ExampleLugubrious Pete's Delicatessen

In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

First, to set the scene:

```
"Lugubrious Pete's Delicatessen"
```

```
The Supermarket is west of the Delicatessen Counter.
Lugubrious Pete is in the Delicatessen. "Lugubrious
Pete, dolefully slicing meats and cheeses, serves at the
counter." Alice, Beth, Gemma, Delia, and Eliza are
women in the Supermarket.
```

```
The deli queue is a list of objects that varies.
```

Two processes compete here: one that fills the queue, the other which will empty it. The first process is the one which brings shoppers in to the counter, joining the back of the queue, which is where "add ... to ..." puts new entries by default:

```
Every turn when a woman is in the Supermarket and a
random chance of 2 in 3 succeeds (this is the customer
arriving rule):
    let the customer be a random woman in the
Supermarket;
    now the customer is in the Delicatessen;
    if the player is in the Supermarket, say "[The
customer] walks into the Delicatessen.";
    if the player is in the Delicatessen, say "[The
customer] comes in from the Supermarket, and [if the
number of entries in the deli queue is 0]can't believe her
luck. The counter is free![otherwise]resignedly queues
behind [the deli queue].";
    add the customer to the deli queue.
```

The competing process is the one which serves shoppers and thus gets rid of them again: unfortunately, it is slower. But Pete is fair if inefficient, and serves the customers in strict order of arrival. Each served customer is removed from the front of the list, and the others therefore all move up a place.

```
Every turn when the number of entries in the deli queue
```

```
is not 0 and a random chance of 1 in 3 succeeds (this is
the customer being served rule):
    let the customer be entry 1 of the deli queue;
    if the player is in the Delicatessen, say "Pete gives a
droopy expression as he serves [customer], who
nevertheless brightens and leaves.";
    if the player is in the Supermarket, say "[customer]
emerges cheerfully from the Delicatessen Counter, and
goes about her regular shopping.";
    now the customer is in the Supermarket;
    remove entry 1 from the deli queue.
```

```
Instead of waiting in the Delicatessen when the number
of entries in the deli queue is not 0, say "Time passes,
for [deli queue] quite as much as for yourself."
```

```
Test me with "wait / wait / wait / east / wait / wait / wait /
wait / wait".
```

That completes the example, but here is a variation to show that queues need not empty from the front. The Deli already looks a pretty sexist establishment, with the customers all being women, but it is about to get a whole lot worse:

```
Modesty is a kind of value. The modesties are positively
prim, buttoned up, modest, flirty, revealing and
downright immodest. Every woman has a modesty.
Alice is positively prim. Beth is downright immodest.
Gemma is modest. Delia is flirty. Eliza is revealing.
```

We could then rewrite the service rule like so:

```
Every turn when the number of entries in the deli queue
is not 0 and a random chance of 1 in 3 succeeds (this is
the customer being served rule):
    let Pete's preference be the deli queue;
    sort Pete's preference in reverse modesty order;
    let the customer be entry 1 of Pete's preference;
    let the first in line be entry 1 of the deli queue;
    if the player is in the Delicatessen, say "[if the
customer is the first in line]Pete gives a droopy
```

```
expression as he serves [the customer], who
nevertheless brightens and leaves.
[otherwise]Outrageously, Pete scans the queue, notices
[the customer] in her [modesty of the customer] clothes,
and serves her next, while [the first in line] glares at
him.";
    if the player is in the Supermarket, say "[The
customer] emerges cheerfully from the Delicatessen
Counter, and goes about her regular shopping.";
    now the customer is in the Supermarket;
    remove the customer from the deli queue.
```

It is now heartbreakingly difficult for Alice to obtain her sliced chorizo sausage.

---

176 **Example** **Strictly Ballroom** *WI*

People who select partners for dance lessons each turn.

Many simple repetitions can effectively be done with a "now..." instruction: it is quicker to say

```
now every person is angry
```

than

```
repeat with offended party running through people:
    now the offended party is angry.
```

Repeat comes in handy when we have something a bit more complicated to do with each item:

```
"Strictly Ballroom"

A person can be alert or occupied. A person is usually
alert.
```

```
When play begins:
    now the player is occupied.


Dance is a kind of value. The dances are waltzes,
polkas, cha-chas, charlestons, fox-trots, tangos,
lambadas, and two-steps.


The current round is a dance that varies.


Manner is a kind of value. The manners are swiftly,
primly, energetically, suavely, seductively, elegantly,
and badly.


Every turn: now the current round is a random dance.


Every turn:
    repeat with dancer running through people who are
not the player:
        if dancer is alert:
            now dancer is occupied;
            let partner be a random alert person who is not
the dancer;
            if partner is a person:
                now partner is occupied;
                say "[The dancer] [the current round][if a
random chance of 1 in 5 succeeds] [a random manner]
[end if] with [partner]. ";
            otherwise:
                say "[paragraph break][The dancer] is forced
to be a wallflower. Poor [dancer]. ";
    say paragraph break.
```

Notice we did not say "repeat with dancer running through alert people who are not the player...". This is because Inform would draw up a list of alert people at the beginning of the repeat, and not take into account which people became occupied partway through the repetition. If we

want to make sure that each person dances only with one other person, we have to continue checking alertness each time we run through the repetition.

After all the partners are assigned, we can set up for the next turn by making everyone alert again, and for this we do not need "repeat":

```
Every turn: now every person is alert; now the player is occupied.

Before doing something to someone: now the noun is occupied.

Before doing something when the second noun is a person: now the second noun is occupied.

Instead of doing something to someone: say "You successfully distract [the noun]."

The Pacific Ballroom is a room. "A rather utilitarian space at the moment, since this is a class and not a party." Timmy, Tommy, Joey, George, Mary, Martha, Yvette, McQueen, Linus, and Patricia are people in the Pacific Ballroom.

Test me with "z / ask linus about blanket / z / z".
```

Social dynamics in which groups of people form and circulate during a party.

To start with, let's understand "room" to mean "a group of people talking".

These groups can grow and shrink as people come and go, so we'll want to name and rename them; and we're also going to need some rules to motivate people moving around, and a description to narrate how they behave when we're with them.

```
"Emma"

by the banquet table is a room. at the corner is a room.
next to the doorway is a room. by the window is a room.

Social clump is a kind of value. The social clumps are
vacancy, lone person, couple, cluster, group.

A room has a social clump. Understand the social clump
property as describing a room.

Before printing the name of a room:
    say "a [social clump] ".

After looking:
    assign clumping;
    say "Elsewhere in the room, you can see [the list of
rooms which are not the location]."

Understand "go to [any room]" as joining. Joining is an
action applying to one visible thing. Carry out joining:
move player to the noun. Report joining: do nothing.

Understand "examine [any room]" as looking toward.
Looking toward is an action applying to one visible thing.
Carry out looking toward a room: say "In that direction
```

you see [a list of other people in the noun]."

When play begins: assign clumping. Every turn: assign clumping.

To assign clumping:
    repeat with space running through rooms:
        now the social clump of the space is vacancy;
        if the space contains exactly 1 person, now the social clump of the space is Lone person;
        if the space contains exactly 2 people, now the social clump of the space is Couple;
        if the space contains more than 2 people and the space contains fewer than 5 people, now the social clump of the space is cluster;
        if the space contains more than 4 people, now the social clump of the space is group.

The room description heading rule is not listed in the carry out looking rules.

A person has a number called longevity. The longevity of a person is usually 0. A person can be active or passive.

Definition: a person is other if it is not the player.

Every turn:
    repeat with mover running through other people:
        now the mover is active;
        increment the longevity of mover;
        if longevity of mover is greater than 3 or the mover is bored:
            assign value of spaces for the mover;
            let destination be the nicest room;
            if the destination is not the location of the mover:

if the player can see the mover, say "[The mover] makes excuses and drifts off to join [the destination].[paragraph break]";
move the mover to the destination;
now the mover is complacent;
now the longevity of the mover is 0;
if the player can see the mover, say "[The mover] wanders over.[paragraph break]";
assign clumping;
now mover is passive.

A room has a number called attractiveness.

Definition: a room is nice if its attractiveness is 1 or more.

To assign value of spaces for (mover - a person):
repeat with space running through rooms:
now attractiveness of the space is 0;
repeat with figure running through people in the space:
if the mover is bored, decrease attractiveness of the space by 2;
if the mover likes the figure, increment attractiveness of the space;
if the mover dislikes the figure, decrement attractiveness of the space;
if the mover desires the figure, increase attractiveness of the space by 2.

Liking relates various people to various people. The verb to like means the liking relation.

Disliking relates various people to various people. The verb to dislike means the disliking relation.

Attraction relates various people to various people. The verb to desire means the attraction relation.

Mr Weston, Mr Woodhouse, Mr Elton, Mr Knightley, and Frank Churchill are men. Mrs Weston, Mrs Bates, Miss Bates, Harriet Smith, Emma Woodhouse, and Jane Fairfax are women.

Harriet Smith likes Mr Elton. Harriet Smith desires Mr Elton. Harriet Smith likes Emma Woodhouse.

Mr Elton desires Emma Woodhouse.

Emma Woodhouse likes Harriet Smith and Mr Knightley. Emma Woodhouse dislikes Jane Fairfax.

Mr Knightley likes Emma Woodhouse, Mr Weston, and Mrs Weston. Mr Knightley desires Emma Woodhouse.

Jane Fairfax desires Frank Churchill. Jane Fairfax likes Frank Churchill.

Frank Churchill desires Jane Fairfax and Emma Woodhouse. Frank Churchill likes Jane Fairfax.

Miss Bates likes Jane Fairfax, Emma Woodhouse, and Mrs Bates.

Mr Weston likes Frank Churchill, Emma, Knightley, and

```
Mrs Weston.


Mrs Weston likes Frank Churchill, Emma, Knightley, and
Mr Weston.


Mrs Bates likes Miss Bates.


A person can be complacent or bored.


When play begins:
    repeat with character running through other people:
        let space be a random room;
        move character to space.
```

And now we use writing a paragraph about... to describe character behavior in groups, when we join them:

```
Rule for writing a paragraph about Frank Churchill:
    if the location contains a woman (called flirt) who is
desired by Frank:
        say "[Frank Churchill] is talking with great
animation and slightly more than becoming warmth to
[the flirt][if an unmentioned other person is in the
location], while [the list of unmentioned other people in
the location] look on with varying degrees of amusement
or irritation[end if].";
        repeat with character running through people in the
location:
            if the character is not Churchill and the character
is not the flirt, now the character is bored.


Rule for writing a paragraph about Mr Elton:
    if the location contains an unmentioned woman
(called flirt) who is desired by Elton:
        say "[Mr Elton] hangs on the sleeve of [the flirt],
offering an assortment of studied gallantries that make
```

```
you wonder about his good sense.";
        repeat with character running through people in the
location:
            if the character is not Elton and the character is
not the flirt, now the character is bored.


Rule for writing a paragraph about Harriet Smith:
    if the location contains Emma and Emma is
unmentioned:
        say "[Harriet] and [Emma] are conversing in low
tones -- Harriet, apparently, being too shy to speak so
that everyone can hear her."


Rule for writing a paragraph about Mr Knightley:
    if the location contains an unmentioned man (called
the listener) who is not Mr Knightley:
        say "[Mr Knightley] is speaking with [the listener]
about agricultural matters.";
        now the listener is complacent.


Rule for writing a paragraph about Miss Bates:
    say "[Miss Bates] is giggling about the weather[if an
unmentioned other person is in the location]. This does
not seem to compel the interest of [the list of
unmentioned other people in the location][end if].";
    repeat with character running through people in the
location:
        if the character is not Miss Bates and character is
not Mrs Bates, now the character is bored.
```

Since this is just an example, we'll stop here, but there's no reason we couldn't write such paragraphs for everyone.

```
Test me with "z / z / z / look / x corner / x doorway / x
window / x table / go to the table".
```

ExampleHappy Hour

Often it is best to have an entire paragraph about the characters present in a room, but suppose we're narrating a large party with a lot of people moving around. In that case, it might be better to list everyone together, then add a few salient details by way of follow-up, like this:

```
"Happy Hour"

Before listing nondescript items:
    say "You can see [a list of people who are marked for
listing] here. ";
    repeat with named party running through people:
        now the named party is not marked for listing;
    let count be the number of visible other people who
are carrying something;
    if count is 0:
        say paragraph break;
        continue the action;
    let index be count;
    repeat with holder running through visible other
people who are carrying something:
        if index is count, say "[The holder]";
        otherwise say "[the holder]";
        say " has [a list of things carried by the holder]";
        decrement index;
        make delimiter index of count.
```

The next part could be simpler, but for rigor we will write it in such a way that it will work whether or not the serial comma is in use. This requires some extra work.

```
To make delimiter (index - a number) of (count - a
number), continuing or halting:
    if index is 0:
        if continuing, say ". [run paragraph on]";
        otherwise say ".";
    otherwise if index is 1:
        if count is 2, say " and ";
        otherwise say "[optional comma] and ";
    otherwise:
```

```
        say ", ".


To say optional comma:
    if the serial comma option is active:
        say ",".
```

And now the scene:

```
The Banquet Hall is a room. "A large cheery banner
over the door (which, incidentally, vanishes when you
approach it) reads: HELLO NEW INDUCTEES!
WELCOME TO THE AFTERLIFE!"


Fred, George, and Larry are men in the Banquet Hall.
Fred carries a dry martini. Larry carries a cream puff.
Matilda and Louise are women in the Banquet Hall.


Definition: a person is other if it is not the player.


Every turn:
    let wanderer be a random other person;
    let place be the holder of the wanderer;
    let next place be a random room adjacent to the
place;
    let the way be the best route from the place to the
next place;
    try the wanderer going the way.


The Kitchen is west of the Banquet Hall. "Dominated by
a pile of dirty plates which you imagine it will be
someone's privilege to wash, later." Vanessa is a
woman in the Kitchen. Vanessa carries a tray. On the
tray is a salmon roll. The roll is edible.
```

Test me with "z / look / g / g / g ".

Chapter 8: Vehicles, Animals and Furniture

Contents of *The Inform Recipe Book*

Chapter 7: Other Characters

Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Indexes of the examples

§8.1. Bicycles, Cars and Boats

The vehicle kind in Inform refers to an object which can carry at least one person, but is small enough to fit into a single location:

```
In the Garden is a vehicle called the motor mower.
```

We can then apply different rules to a player going somewhere on foot or in the vehicle. **Peugeot** (a bicycle) is an easy example; **No Relation** (a car) adds an ignition switch to the vehicle; **Straw Boater** (a motorboat) gets around areas of lake where travel on foot is not just slower but impossible.

Hover (a sci-fi "hover-bubble") changes the appearance of the landscape when it is seen from inside the vehicle.

See **Ships, Trains and Elevators** *for larger conveyances*

Start of Chapter 8: Vehicles, Animals and Furniture

Back to Chapter 7: Other Characters: §7.16. Social Groups

Onward to §8.2. Ships, Trains and Elevators

Example 27: **Peugeot** A journey from one room to another that requires the player to be on a

vehicle.

Example 104: ★ **No Relation** A car which must be turned on before it can be driven, and can only go to roads.

Example 49: ★★ **Straw Boater** Using text properties that apply only to some things and are not defined for others.

Example 29: ★★★ **Hover** Letting the player see a modified room description when he's viewing the place from inside a vehicle.

---

§8.2. Ships, Trains and Elevators

This section covers vehicles whose interior consists of at least one entire room. Moving into this room constitutes boarding: there is then some pause while it travels: moving out again disembarks at a different location. The only complication arising is how the player controls the journey (by talking to someone? by pressing buttons? by steering?).

The Unbuttoned Elevator Affair provides the simplest possible whole-room vehicle, which ferries between two fixed points. If you are at one of these, it must be the other one you want to get to, so there is no need for controls.

Dubai is a much more elaborate elevator, with many possible destinations, chosen using buttons inside the elevator.

Empire simulates a train journey. Here there are no controls as such, but the train passes through a sequence of stops spaced apart in time, so the player chooses an exit by getting out at the right moment.

On a very large, slowish craft such as a cruise liner, we are not so much travelling in a vehicle: it's more as if we are visiting a whole building, which becomes our world for the (probably long) duration of the journey. The liner steers around in long, slow curves, changing its orientation in the water, so that (if we think of "north" as a strictly magnetic matter, anyway) north is constantly rotating: something we don't notice on board because our own reference points, provided by the ship itself, stay fixed relative to ourselves. Because of this, some ships in IF are navigated using FORE, AFT, PORT and STARBOARD directions rather than NORTH, EAST, SOUTH and WEST: see **Fore**.

See **Bicycles, Cars and Boats** *for smaller conveyances*

Start of Chapter 8: Vehicles, Animals and Furniture
Back to §8.1. Bicycles, Cars and Boats
Onward to §8.3. Animals

Example 9: ★ **The Unbuttoned Elevator Affair** A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

Example 145: ★★ **Empire** A train which follows a schedule, stopping at a number of different locations.

Example 42: ★★★ **Fore** Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

Example 271: ★★★ **Dubai** An elevator which connects any of 27 floors in a luxury hotel.

§8.3. Animals

Animals exhibit a wide range of behaviour: much of the chapter on Other Characters applies just as well to animals as to human beings, with the exception of the material on conversation. But two examples here, both fairly simple, show how a fairly convincing domestic pet can be achieved simply by reacting to certain events going on nearby: **Feline Behavior** (a cat) and **Today Tomorrow** (a dog).

Fido provides a dog which the player can re-name at will.

For animals that we can sit on and ride - a camel or a horse, say - we may want to use the Rideable Vehicles extension by Graham Nelson, which also provides a rideable animal kind.

See **Liveliness** *for pets that change what they're doing every time the player looks*

See **Bags, Bottles, Boxes and Safes** *for a cat that eats food put in its container*

Start of Chapter 8: Vehicles, Animals and Furniture

Back to §8.2. Ships, Trains and Elevators

Onward to §8.4. Furniture

Example 406: **Feline Behavior** A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

Example 420: **Fido** A dog the player can name and un-name at will.

Example 99: **Today Tomorrow** A few notes on "In the presence of" and how it interacts with concealed objects.

---

§8.4. Furniture

Most domestic furniture consists of supporters and containers of one size or another. This means that the simplest furniture needs no elaborate instructions:

> The candlestick is on the dining table. The dining table is fixed in place.

> The silver salt cellar is on the serving trolley. The serving trolley is pushable between rooms.

The pillow is on the bed. The bed is enterable and fixed in place.

The examples below are therefore mostly ways to get around the usual restrictions on containers (that they only have one interior) and supporters (that they cannot simultaneously be containers as well).

Yolk of Gold provides a set of drawers, that is, a container with multiple interiors.

U-Stor-It provides a way to have containers with a lid which is also a supporter.

Swigmore provides a supporter which holds up the player, but has no top surface as such, and cannot hold up anything else. **Kiwi** demonstrates a kind of high shelf, whose objects cannot be seen or used unless the player stands on a ladder.

Princess and the Pea shows how a pile of supporters, each on top of the last, could be managed.

Tamed demonstrates furniture large enough to get inside, or on top of.

Circle of Misery demonstrates a conveyor belt, which can hold multiple items but only brings one of them within the player's reach at a time.

See **Position Within Rooms** *for a box that can be positioned and used as a stepping stool*

See **The Human Body** *for letting the player take different postures on furniture or on the floor*

See **Room Descriptions** *for tables and other furniture whose content listing is suppressed or modified in a room description*

See **Entering and Exiting, Sitting and Standing** *for making the player automatically rise from a seat before leaving the room*

See **Clocks and Scientific Instruments** *for a grandfather clock*

See **Kitchen and Bathroom** *for a mirror the player can look into*

Start of Chapter 8: Vehicles, Animals and Furniture

Back to §8.3. Animals

Onward to §8.5. Kitchen and Bathroom

Example 13: ★ **Tamed**
Examples of a container and a

supporter that can be entered, as well as nested rooms.

Example 173: ★ **Princess and the Pea** The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

Example 361: ★ **Kiwi** Creating a raised supporter kind whose contents the player can't see or take from the ground.

Example 432: ★ **Circle of Misery** Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

Example 83: ★★ **Yolk of Gold** Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

Example 397: ★★ **Swigmore U.** Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

Example 58: ★★★ **U-Stor-It** A "chest" kind which consists of a container which has a lid as a supporter.

---

§8.5. Kitchen and Bathroom

Before implementing elaborate mechanisms to handle plumbing, we should pause to ask ourselves: how much of this do we need? Is it really necessary to simulate the complete set of fixtures and fittings?

This turns out to be a little tricky to do, and also rather dull to set out. The example **Modern Conveniences** was actually written as a demonstration of how an extension to Inform might be written to provide a general "kitchens and bathrooms service" for writers, but it contains a nice implementation well worth borrowing. The idea is to provide a "kitchen" kind of room and a "bathroom" kind of room. All kitchens created automatically contain standard kitchen appliances: fridge, freezer, sink with taps, counters, cabinets, and a stovetop with built-in oven. Similarly, all bathrooms will have sinks, baths, cabinets, and toilets, and respond to some standard interactions.

Another common feature of bathrooms is a mirror: **Versailles** demonstrates how to create a simple one.

Start of Chapter 8: Vehicles, Animals and Furniture

Back to §8.4. Furniture

Onward to Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics: §9.1. Food

Example 75: ★★ **Versailles** A mirror which will reflect some random object in the room.

Example 449: ★★ **Modern Conveniences** Exemplifying the kind of source we might use in writing extensions for kitchen and bathroom appliances.

Examples from Chapter 8: Vehicles, Animals and Furniture

Start of this chapter

Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Indexes of the examples

★ **ExamplePeugeot**
A journey from one room to another that requires the player to be on a vehicle.

Let's say that our protagonist is about to flee . Obviously, he can't make the journey on foot; he needs transportation.

```
"Peugeot"

Include Rideable Vehicles by Graham Nelson.

The Lot is a room. The ten-speed bike is a rideable
vehicle in the Lot.
```

We make the ten-speed bike a rideable vehicle because we want to say that the player is on it rather than in it. Then our other room:

```
Cambridge is east of the Lot.
```

And now we borrow from the Actions chapter to prevent travel without the proper equipment:

```
Instead of going to Cambridge when the player is not on
the ten-speed bike:
    say "It's a long journey to Cambridge: you'll never
make it on foot."

After going to Cambridge:
    say "You begin pedalling determinedly.";
    continue the action.

Test me with "e / get on ten-speed bike / e".
```

Example **No Relation**

We need to designate certain rooms as roads. Since the status of being a road will not change during play, we do this with a kind:

"No Relation"

A road is a kind of room. Definition: a room is offroad if it is not a road.

Instead of going by a vehicle (called the auto) to somewhere offroad:
    say "You can't drive [the auto] off-road."

Trafalgar Square is a road. "The Square is overlooked by a pillared statue of Admiral Lord Horatio Nelson (no relation), naval hero and convenience to pigeons since 1812."

The National Gallery is north of Trafalgar Square. The Strand is east of Trafalgar Square. The Strand is a road.

The car is a vehicle in Trafalgar Square. The ignition is a device. The ignition is part of the car. Instead of going by the car when the ignition is switched off: say "The ignition is off at the moment." Instead of switching on the car, try switching on the ignition. Instead of switching off the car, try switching off the ignition.

Test me with "get in car / n / e / turn on car / n / e / get out / w / n / s / e / get in car / turn off car / w / turn on

```
ignition / w".
```

(In the course of the writing of Inform 7, much of Trafalgar Square was pedestrianised, making this example already out of date.)

A further technical note: notice "going by a vehicle" in the above rule, rather than "going by something". A rule such as "Instead of going by something..." will be matched whenever the player tries to go some direction while in an enterable object, whether or not that object is actually capable of movement. This is sometimes useful, but in this case we want the warning to apply only when the player is in a vehicle; if we added Trafalgar Square's statue bases to the scenario, we would not want

```
You can't drive the pedestal off-road.
```

So we restrict the rule to "Instead of going by a vehicle..."

---

49. Example Straw Boater *WI*

Using text properties that apply only to some things and are not defined for others.

Sometimes we like to give properties to kinds of thing, but not fill them in in all cases. For instance, we might have vehicles that optionally make noise, and those might have a "movement sound".

All properties have a default value, which we can find by looking in the Kinds tab of the index. This is what the property will be set to automatically, if we do not change it ourselves. In the case of a text property, that is ""; so for instance we might use our movement sound thus:

```
"Straw Boater"

Boathouse is a room. "A boathouse circa 1915, which --
though in poor repair -- still suggests Sunday afternoon
jaunts taken by women in white gowns and men in straw
hats."
```

North of the Boathouse is the Shallow Water. The description of Shallow Water is "Just south is the boathouse, and beyond it are trees and the marble terrace of the house above. The water deepens to the north."

North of Shallow Water is Deep Water. The description of Deep Water is "From here the boathouse has dwindled invisibly to the south, and you have a broad panorama of the shoreline, all the way down to the Skeleton Point Lighthouse in the southeast."

A vehicle has some text called the movement sound. The sailboat and the motorboat are vehicles in the Boathouse. The movement sound of the motorboat is "VRRRROOOMMMM..." Understand "boat" as the sailboat. Understand "boat" as the motorboat.

Note that we haven't given the sailboat any movement sound at all.

After going somewhere by a vehicle (called cart):
    if the movement sound of the cart is not "", say "[the movement sound of the cart][paragraph break]";
    continue the action.

Instead of exiting when the player is in a vehicle and the location is not the Boathouse:
    say "You're not dressed for a swim."

Instead of going somewhere when the player is not in a vehicle:
    say "You'd rather not try to make this journey by swimming alone."

Test me with "n / get in sailboat / n / get out / s / get in motorboat / n / n".

 ExampleHover *WI*

Letting the player see a modified room description when he's viewing the place from inside a vehicle.

Suppose we want the player to see a modified room description when he's viewing the place from inside a vehicle. There are several conceivable ways of doing this; the example here shows a rather advanced way, but is very flexible and will let us write all sorts of special cases.

"Hover"

Use full-length room descriptions.

Emerald City is a room. "All the buildings are spires and none of them have doors." The Vast Desert is west of Emerald City. "[if the player is in a vehicle]Outside, a[otherwise]A[end if] trackless waste stretches as far as the eye can see in every direction."

The hover-bubble is a vehicle in the Emerald City. "Your hover-bubble awaits." The description is "The hover-bubble is a clear globe-shaped vehicle capable of transporting you anywhere you could walk, but faster." Understand "bubble" as the hover-bubble. The hover-bubble contains a chocolate wrapper and a parking ticket.

Here's the tricky part, which relies on material from the chapters on Activities and Rulebooks:

```
The container interior rule is listed before the room
description body text rule in the carry out looking rules.


This is the container interior rule:
    if the actor is the player and the player is in an
enterable thing (called current cage), carry out the
describing the interior activity with the current cage.


Describing the interior of something is an activity.
```

Now we've done that, we can write a "rule for describing the interior" of something, which will print whatever we like:

```
Rule for describing the interior of the hover-bubble:
    say "The hover-bubble is transparent, but tints
everything outside very faintly lavender."
```

In fact, as a special refinement, we could even say:

```
Rule for describing the interior of the hover-bubble when
the hover-bubble contains more than one thing:
    say "The hover-bubble is transparent, but tints
everything outside very faintly lavender. Beside you you
can see [a list of other things in the hover-bubble]."


Definition: a thing is other if it is not the player.


Rule for listing nondescript items of the hover-bubble
when the player is in the hover-bubble: do nothing.


Test me with "get in bubble / look / west / take all / look /
get out / east".
```

And now anything that's beside us in the vehicle will be described during

that first paragraph, rather than later on.

Example **The Unbuttoned Elevator Affair** *WI*

A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

This is very simple. The interior of the elevator is a single room, but which is mapped east of both of its termini. The reverse map connection, west from the elevator, can only go to a single room, and that's what determines which floor the elevator is on.

```
"The Unbuttoned Elevator Affair"

UNCLE Headquarters is a room. "The steel nerve-center of the free world's battle against the Technological Hierarchy for the Removal of Undesirables and the Subjugation of Humanity. Being against technology, we have only a very simple elevator to the east."

Del Floria's Tailor Shop is a room. "Only trained anti-THRUSH agents recognise the booth in the east wall as a secret elevator."

The Secret Elevator is east of UNCLE Headquarters.
The Secret Elevator is east of Del Floria's Tailor Shop.

After going to the Secret Elevator:
    say "The doors automatically close, there is a rush of motion, and they open again.";
    if UNCLE Headquarters is mapped west of the Secret Elevator, now Del Floria's Tailor Shop is mapped west of the Secret Elevator;
    otherwise now UNCLE Headquarters is mapped west
```

```
of the Secret Elevator;
    continue the action.


Test me with "east / west / east / west".
```

ExampleEmpire

A train which follows a schedule, stopping at a number of different locations.

Suppose we want to have a train which, at fixed times, arrives at and leaves stations. It should be possible for the player to get on and off the train when it is stopped, but not while the train is in motion.

```
"Empire"


The Empire Builder Train is a room. The Train has a
room called the station. The station of the Train is
Seattle.


The description of the Empire Builder is "One of the
(relatively) plush long-distance Amtrak trains. You're in
a two-story car with toilets and a cafe at one end, not
having sprung for a sleeper.


[if the station of the Train is the train]Outside the window
there is rapidly-passing countryside.[otherwise]Through
the windows you can see the [station of the Train] train
station.[end if]"


Instead of exiting when the player is in the Train:
    if the station of the Train is the Train:
        say "The train is not stopped at a station." instead;
```

```
    otherwise:
        move the player to the station of the train instead.


Before going outside when the player is in the Train:
    try exiting instead.


Before going inside when the player is in the station of
the Train:
    move the player to the Train instead.


Seattle, Edmonds, Everett, Wenatchee, and Spokane
are rooms. The description of a room is usually "The
scenic train station of [the location][if the location is the
station of the train].


The pompously-titled Empire Builder train is pulled up
here, soon to continue its journey towards Chicago[end
if]."
```

And now our schedule for the train -- somewhat truncated, admittedly, since the full three-day journey from Seattle to Chicago is a bit long even for an ambitious example.

```
At 4:45 PM:
    if the player is in the train or the player is in the
station of the train, say "The train pulls out of [the station
of the Train]!";
    now the station of the Train is the Train.


At 5:10 PM:
    now the station of the Train is Edmonds;
    if the player is in the train or the player is in the
station of the train, say "The train pulls into Edmonds
and comes to a stop."
```

```
At 5:17 PM:
    if the player is in the train or the player is in the
station of the train, say "The train pulls out of [the station
of the Train], running north along the shore towards
Everett.";
    now the station of the Train is the Train.

At 5:39 PM:
    now the station of the Train is Everett;
    if the player is in the train or the player is in the
station of the train, say "The train arrives in scenic
Everett, WA: the last stop before it turns east and heads
over the mountains."

At 5:44 PM:
    if the player is in the train or the player is in the
station of the train, say "The train pulls out of [the station
of the Train] and turns east.";
    now the station of the Train is the Train.

At 8:39 PM:
    if the player is in the train or the player is in the
station of the train, say "In darkness the train rolls into
Wenatchee; which is just fine, considering that there is
nothing to see here at all.";
    now the station of the Train is Wenatchee.

At 8:44 PM:
    if the player is in the train or the player is in the
station of the train, say "The train pulls out of [the station
of the Train] and continues east through the darkness
towards Spokane.";
    now the station of the Train is the Train.
```

Playing this out would of course require near inhuman patience. Let's set things up so that the player at least doesn't have to wait too long for his first departure:

```
The time of day is 4:43 PM.
```

...and provide fair warning of how slowly time is elapsing.

```
When play begins:
    now the right hand status line is "[time of day]".
```

```
Test me with "out / in / z/ z / z / out / z / z / z / z / z / z / z
/ z / z / z / z / z / z / z / z / z / z / z / z / z / z / z / out / in
/ out / z / z".
```

```
Test more with "out / z/ z / z / z / z / z / z / z / z / z / z / z /
z / z / z / z / z / z / z / z / z / z / z / z / z / z / z / z".
```

ExampleFore

Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

Suppose we want to understand shipboard directions, but only when the player is aboard a vessel.

```
"Fore"
```

```
Section 1 - Procedure
```

```
The starboard is a direction. The starboard has opposite
port. Understand "s" as starboard when the location is
nautical.
```

```
The port is a direction. The port has opposite starboard.
Understand "p" as port when the location is nautical.
```

```
The fore is a direction. The fore has opposite aft.
Understand "f" as fore when the location is nautical.


The aft is a direction. The aft has opposite fore.
Understand "a" as aft when the location is nautical.


Does the player mean going a nautical direction when
the location is nautical: it is very likely.


Index map with fore mapped as north. Index map with
aft mapped as south. Index map with port mapped as
west. Index map with starboard mapped as east.
```

And we can even add new ways to talk about the ways things are mapped, borrowing from the Relations chapter. The following will allow us to use "is abaft of" as well as "is aft of":

```
[The verb to be abaft of means the mapping aft relation.]
```

Now, to prevent the player from using NORTH onboard ship, or AFT on land:

```
A room can be nautical or earthbound. A room is usually
not nautical. A direction can be nautical or earthbound.
A direction is usually not nautical. Starboard, port, fore,
aft, up, down, the inside and the outside are nautical.


Before going a nautical direction when the location is
not nautical, say "Nautical directions can only be used
on board ship."


Before going an earthbound direction when the location
```

is nautical, say "Compass directions make no sense on board ship, but you can use [list of nautical directions] instead." instead.

Section 2 - Scenario

The Fish Room is aft of the Spirit Room. Starboard of the Fish Room is the After Powder Magazine. The Bread Room is aft of the After Powder Magazine.

The Fish Room, the Spirit Room, the Bread Room, and the After Powder Magazine are nautical.

The description of the Fish Room is "Absurd quantities of salt fish are kept here, and periodically visited by the cook or someone serving him. It is otherwise an unexceptional little chamber, so far below the waterline that there are no portholes and no external light of any kind. [paragraph break]A narrow doorway leads forward into the Spirit Room, and the After Powder Magazine is starboard."

The description of the Spirit Room is "Despite its ghostly name, this is little more than a closet down at the very navel of the ship, in which alcohol is kept: both for purifying wounds and for drinking. Under normal circumstances there is a guard posted here at every hour, lest anyone take to raiding the larder. The current absence of the guard marine strikes you as a very bad sign indeed. [paragraph break]The only way out is aft."

The description of the Bread Room is "The Bread Room is not only tiny from side to side and front to back: it is also about half the height of a proper room, and the floor slopes up very steeply with the curve of the hull. [paragraph break]What is kept here would not, on land,

be dignified by the name of bread: it's hard tack, punishing to the teeth, dry on the tongue, and usually a home to weevils before half the journey is done. [paragraph break]More headroom, and access to the rest of the ship, lies fore through the After Powder Magazine."

The description of the After Powder Magazine is "Kept in near darkness because no one with any sense would bring a naked flame down here: when necessary, it can be lit with a single small lantern made of very thick glass and sealed to keep the sparks within. Sacks of powder are passed up into the higher levels of the ship by the scrubby little boys called 'powder monkeys' -- but none such are here now."

Test me with "north / aft / fore".

---

271 **ExampleDubai** *WI*

An elevator which connects any of 27 floors in a luxury hotel.

The problem of implementing an elevator that opens onto a large number of floors often challenges novice interactive fiction authors. It also raises a fundamental design problem -- how to implement a large building in an interesting way. It is tempting to write a hotel with an elevator and innumerable tedious and identical floors just for the sake of realism; in many cases it is a better idea simply to omit any locations that contribute nothing to the story or the play of the game.

In charity, though, let us assume that the author has a legitimate reason for wanting to implement an elevator that opens onto some generic floors. We will go whole-hog, and set this in the world's tallest hotel: Burj al-Arab, Dubai.

"Dubai"

The Burj al-Arab Lobby is a room. "The 202-suite Burj al-Arab - or Tower of the Arabs - stands 321 metres (1,060 feet) high, and floats on its own man-made island. It is shaped like the sail of a boat; just crossing the private bridge to reach this place set you back $55."

(Since our budget did not run to visiting Burj al-Arab, the descriptions place implicit faith in the hotel's website.)

The Assawan Spa is a room. "Treatment rooms, hydrotherapy baths, oriental massage, stand-up solarium, sauna, steam rooms and jacuzzi, two swimming pools, squash court, two fully equipped fitness studios and an aerobics floor. To the south is a shopping area, for those who do not find exercise adequately therapeutic."

The Shopping Area is south of Assawan Spa. "In a setting that would make a poet sigh, you may enjoy the services of (among others) Bulgari, Black Pearl Caviar, Albarajeel Carpet Shop, Abdul Samed Al Qurashi (amber and Arabic perfumes), Dianoor (jewellery), and the Commercial Bank of Dubai."

The Al Falak Ballroom is a room. "A palatial, two-tiered, domed ballroom crowned with a unique crystal chandelier."

Sahn Eddar is a room. "At the base of the world's tallest atrium, the Sahn Eddar restaurant offers light fine fare and Afternoon Tea. At the center, a 32-meter water column leaps toward the roof of the atrium."

Al Mahara is a room. "After the elevator, you must take a three-minute virtual submarine voyage to reach this seafood restaurant. A magnificent oval aquarium, full of sharks, is visible from every table."

Al Iwan is a room. "Middle-eastern food in an environment of dramatic black, red, and gold."

Al Muntaha is a room. "A top-floor restaurant with a magnificent view: the name means the ultimate or the highest, suitable for a place that stands at 200 metres above the Arabian Gulf. It offers modern European cuisine; and just off to the south is the Skyview Bar."

The Skyview Bar is south of Al Muntaha. "'A wonderful location for pre- and post dinner drinks,' claims the hotel's brochure, and certainly you can't fault the view."

The Juna Lounge is a room. "Two humidors that offer one of the largest selections of the finest Havana cigars in Dubai."

The Lift is a room. "This is not a mere elevator: it is an express panoramic lift, traveling six meters a second, and capable of taking you from lobby to the rooftop restaurant in an astonishingly short time. The illuminated number above the door says [current level of the Lift] - though you can make it move by pressing a numbered button."

The Presidential Suite is a room. "Astonishingly, this is not the most elaborate or expensive of suites; there is another, the Royal, upstairs of here."

The Royal Suite is a room. "The brochure described this as 'the last word in luxury,' and you have to admit that it is certainly the last word in [italic type]something[roman type]. A vast carpet of patterned red and gold stretches from you to the sofa; beyond which, in the distance, you make out several bedrooms and bathrooms outfitted in Carrera marble. There is also, of course, a private

cinema."

Table of Floors

floor	level
0	Al Mahara
1	Burj al-Arab Lobby
2	Al Iwan
3	Juna Lounge
4	Sahn Eddar
15	Al Falak Ballroom
18	Assawan Spa
24	Presidential Suite
25	Royal Suite
27	Al Muntaha

The elevator exterior is a backdrop. It is not scenery. The initial appearance of the elevator exterior is "You can enter the elevator here." It is in Generic Floor, Al Mahara, the Lobby, Al Iwan, Juna Lounge, Sahn Eddar, Al Falak, Assawan Spa, Presidential Suite, Royal Suite, and Al Muntaha.

Before entering the elevator exterior, try going inside instead.

Instead of going inside in the presence of the elevator exterior:
    if there is a level corresponding to a floor of the location in the Table of Floors:
        let the present level be the level corresponding to a floor of the location in the Table of Floors;
        now the current level of the Lift is the present level;
    otherwise:
        now the current level of the Lift is the current level of Generic Floor;
    move the player to the Lift.

The Lift has a number called current level. The current level of the Lift is 1. Instead of going up in the Lift: say "You'll have to select a specific floor; your options range from 0 to 27." Instead of going down in the Lift: try going up instead. The Lobby is outside from the Lift.

Before going outside in the Lift:
    if there is a floor corresponding to a level of the current level of the Lift in the Table of Floors:
        let the other place be the floor corresponding to a level of the current level of the Lift in the Table of Floors;
        move the player to the other place instead;
    otherwise:
        now the current level of the Generic Floor is the current level of the Lift;
        move the player to the Generic Floor instead.

The Generic Floor is a room. The Generic Floor has a number called current level. The printed name of the Generic Floor is "Floor [current level of the Generic Floor]". "A long hallway between suites, some of which run up to $15,000 a night."

Understand "push [number]" as pressing button.
Understand "push [number] button" as pressing button.
Understand "push button [number]" as pressing button.
Pressing button is an action applying to one number.

Check pressing button:
    if the player is not in the Lift, say "You cannot control the express panoramic lift unless you are yourself inside." instead;
    if the number understood is the current level of the Lift, say "The lift pings politely and reopens its doors, since you are already on floor [number understood]." instead;
    if the number understood is greater than 27, say "There are only 27 floors." instead;
    if the number understood is less than 0, say "You cannot go below the ground floor in this elevator." instead.

```
Carry out pressing button:
    now the current level of the Lift is the number
understood;
    say "You press button [the number understood]. The
lift whirs into action and moves to the correct level."
```

```
Test me with "press 3 / in / press button 3 / look / out / in
/ press 27 / out / s / in / n / in / press 15 / out / in / press
18 / out / s / in / n / in / press 4 button / out".
```

This will all work very well, unless the player has portable objects; in that case, anything he drops on the Generic Floor will be there every time he goes back -- whether it's masquerading as Floor 6 or Floor 23. There are there are two ways round this -- (i) the cheeky way. When we drop something, the unobtrusive yet ever-vigilant maids pick it up and take it down to the Foyer's lost property office; and (ii) the super-duper way, in which things are moved out of play but with their floor numbers remembered, so that the scenario can be reconstructed each time. (i) is probably in fact the more true-to-life, considering the hotel's boasts about its service, but we will demonstrate both methods.

Here is the version with vigilant maids:

```
The player carries a shopping bag. In the bag are a
diamond necklace, a small rug, and a jar of caviar.
```

```
Carry out pressing button:
    now every portable thing enclosed by the Generic
Floor is in the Office.
```

```
The Office is south of the Lobby. "Here the maids collect
everything abandoned by careless guests." The printed
name of the Office is "Lost and Found Office".
```

```
Test maids with "in / press 6 / out / drop all / in / out / in /
press 23 / out / in / press 1 / out / s / get all".
```

Notice that we tie the maid service to the pressing of the lift button, so that if the player just goes into the lift and comes out again, the maids will not have had a chance to clear his possessions.

Alternatively:

```
The player carries a shopping bag. In the bag are an
evening gown, a bolero jacket, and tickets to the Wild
Wadi Animal Park.
```

```
Carry out pressing button when something is in the
Generic Floor:
   repeat with item running through portable things in
the Generic Floor:
      clear the item;
   repeat with item running through portable things
enclosed by the Generic Floor:
      clear the item.
```

The "enclosed by" line clears even things left on, say, small un-portable side-tables or whatever; but because we do "in" first, we make sure to move any containers or supporters undisturbed. The next bit could be more tidily incorporated into our previous "before going outside" rule, but since we are writing this code to be optionally pasted onto the end of the first bit, we'll express the rule separately:

```
Before going outside in the Lift when something is in
Limbo:
   unless there is a floor corresponding to a level of the
current level of the Lift in the Table of Floors:
      repeat with item running through things in Limbo:
         if the current level of the item is the current level
of the Lift, move the item to the Generic Floor.
```

```
A thing has a number called current level.
```

```
To clear (item - a thing):
   now the current level of the item is the current level of
the Lift;
```

```
move item to Limbo.


Limbo is a room.


Test management with "get all from bag / in / press 22 /
out / drop tickets / in / press 23 / out / drop gown / in /
press 22 / out / get tickets / in / press 23 / out / get
gown".
```

And now we have a situation in which the player's valuables are left untouched wherever in the hotel he happens to abandon them.

Incidentally, this example was almost set in an entirely different location: the largest hotel in the world may some day be the Ryugyong Hotel in Pyongyang, North Korea, with its 105 floors -- but for some years construction halted at the creation of the building's huge concrete shell.

Example **Feline Behavior** *WI*

A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

Suppose we have a cat which is supposed to react to (and destroy) the most interesting thing in its environment. There are several ways we could approach this problem, but for the sake of demonstration, let's have it follow a rulebook to figure out which thing it most wants to interact with. We will then return the chosen object as "the object produced by the cat behavior rules".

```
"Feline Behavior"


The Kitchen is a room. The cat is an animal in the
Kitchen. In the Kitchen is a bowl, a ball of wool, a
newspaper. The bowl contains a quantity of cream.
```

The cat is wearing a silver collar. The description of the cat is "It is wearing [a list of things worn by the cat]."

The player carries a closed openable container called a bag. The bag contains catnip.

The cat behavior rules is a rulebook producing an object.

A cat behavior rule when the cat can touch the catnip:
    say "The cat frolics with the catnip until nothing remains of it.";
    rule succeeds with result catnip.

A cat behavior rule when the cat can touch the cream:
    say "The cat laps up the cream.";
    rule succeeds with result cream.

A cat behavior rule when the cat can touch the ball of wool:
    say "The cat makes the ball of wool into a useless tangle which must be discarded.";
    rule succeeds with result ball.

A cat behavior rule when the cat can touch the newspaper:
    say "The cat bats playfully at the newspaper until all the nasty boring articles are destroyed.";
    rule succeeds with result newspaper.

A cat behavior rule:
    say "The cat looks miffed at the lack of ready entertainment, and glares at you with yellow eyes as though wondering whether your pants leg is good for

```
claw-sharpening.";
    rule fails.



Every turn:
    let the destroyed object be the object produced by the
cat behavior rules;
    if the destroyed object is not nothing:
        now the destroyed object is nowhere;
        say "[line break]Good thing you have no use for
[the destroyed object] yourself.[paragraph break]".



Test me with "z / z / open bag / z / z".
```

We include the if rule succeeded... condition here because nothing will be returned if the cat's search failed (as for instance in the result of the final rule).

Naturally, if we wanted we could equally well ask "if rule failed...".

---

420 ExampleFido *WI*

A dog the player can name and un-name at will.

Suppose we'd like to have a dog which the player is allowed to name himself. We'd like to deal correctly with both

```
>name the dog fido
```

and

```
>name the dog "fido"
```

so we'll also need to strip quotation marks out of the command. We can do this as follows:

"Fido"

The Back Yard is a room.

A dog is an animal in Back Yard. The dog has some text called the nickname. The nickname of the dog is "nothing". Understand the nickname property as describing the dog.

Rule for printing the name of the dog when the nickname of the dog is not "nothing":
    say "[nickname of the dog]"

Naming it with is an action applying to one thing and one topic. Understand "name [something] [text]" as naming it with. Check naming it with: say "You can't name that."

Instead of naming the dog with "nothing":
    now the nickname of the dog is "nothing";
    now the dog is improper-named;
    say "You revoke your choice of dog-name."

Instead of naming the dog with something:
    let N be "[the topic understood]";
    replace the text "'" in N with "";
    now the nickname of the dog is "[N]";
    now the dog is proper-named;
    say "The dog is now known as [nickname of the dog]."

Test me with "name the dog Fido / name the dog Lawrence / look / x lawrence / name Lawrence nothing / look / x lawrence".

Rules about concealment will affect "in the presence of", too. For instance, suppose we have a man with a pocket pet:

"Today Tomorrow"

The Temporary Employment Office is a room. "'Tomorrow's Temporary Workers - Today!' proclaims the logo over the door. The office is divided into two areas, the inner sanctum where you take calls and fiddle with the computer, and the outer area where workers take skill exams and watch inspirational videos ('Earn your way to partial benefits!', 'Vacation days and you!', 'Temping the Tomorrow Way', etc.)."

Maya is a woman in the Office. "Your coworker Maya sits at her own computer, diligently modifying all the [one of]pay rates in the database down from $9.00 an hour to $8.96[or]billing rates in the database up from $25.00 an hour to $25.04[purely at random]." She wears a trenchcoat. She carries a chihuahua. The description of Maya is "[if Maya is wearing the trenchcoat]She's wearing a trenchcoat, in a bizarre bid to keep your boss unaware of the chihuahua in her pocket. 'Because petsitters are really expensive!' she explained to you in an urgent hiss this morning over the coffeemaker. [otherwise]She looks cold. [end if]To all appearances, she is carrying [a list of unconcealed things carried by Maya]."

Maya's computer is scenery in the Employment Office. The description is "You can't see the screen from here, but she was perfectly happy to tell you what she was doing."

Rule for deciding the concealed possessions of someone (called carrier):
    if the particular possession is the chihuahua and the carrier wears the trenchcoat, yes;
    otherwise no.

Instead of eating something in the presence of the chihuahua:
    say "[The chihuahua] yips at you! Maya looks despairingly at [the noun], which is obviously inciting it."

The desk is scenery in the Office. On the desk are a multi-line telephone and a printer. The description of the printer is "Every morning, this instrument of torture spits out a list of the assignments you have to fill - professional, attractive receptionist with level three Excel certification, at $7.05 an hour; that sort of thing. You spend the ensuing three hours trying to meet its tyrannical demands." The description of the multi-line telephone is "Line three blinks urgently at you. You'll pick it up again as soon as you can remember who it was you put on hold."

The player carries a double bacon cheeseburger and a chocolate shake. Understand "milkshake" or "beverage" as the shake. The description of the cheeseburger is "A minor self-indulgence to make up for the fact that you have to work through lunch." The cheeseburger is edible. The shake is edible. The description of the shake is "It has the consistency of spackle and is no longer cold, but there is some chocolatey goodness in it still." Understand "burger" or "hamburger" as the cheeseburger. Instead of drinking the chocolate shake: try eating the shake instead.

Test me with "x maya / x cheeseburger / drink shake / eat cheeseburger".

Unless we somehow get the trenchcoat away from Maya, the chihuahua

will not be in view, and will not intervene in our lunch. All very well for the player character, but not so interesting to the story... To this end, we might add an unfortunate event, courtesy of later chapters:

```
The time of day is 11:45 AM.


At 11:47 AM: say "Your boss pokes his head in,
temporarily free of the round of conference calls that
occupy all his days. 'Maya,' he says. 'Your coat?' He
shakes his head, clucking sadly. 'It doesn't say
professional!' But mercifully Maya manages to take it off
so slowly that he doesn't glimpse her pet before her
phone rings again.";
    now Maya carries the trenchcoat.
```

---

13 ExampleTamed

Examples of a container and a supporter that can be entered, as well as nested rooms.

WI

Within a room, we might have containers and supporters that a player can enter. A chair, stool, table, dais, or pedestal would be an enterable supporter (anything we would describe a person as being "on"); a cage, hammock, or booth would be an enterable container (because we would describe the person as being "inside").

When the player is in or on something, he is able to see the rest of the contents of the room, but a note such as "(in the hammock)" or "(on the poster bed)" is added to the room title when he looks around.

Here is an example to show off the possibilities:

```
"Tamed"


The Center Ring is a room. The cage is in the Center
Ring. A lion is an animal in the cage. The cage is
enterable, openable, transparent, and closed.
```

Notice that we made the cage transparent. Strictly speaking it is not made of transparent materials, but we can see into (or out of) a closed cage due to the gaps between the bars, so that from Inform's point of view a cage behaves much like a large sturdy glass box. (If we really wanted to make a distinction between, say, an airtight container and one with perforations, we could do so, but Inform does not model such nuances by default.) If a container is not transparent, we can see into and out of it only when it is open.

Supporters are a bit more straightforward because there is no circumstance in which they separate the player from the rest of the world:

```
The pedestal is in the Center Ring. It is enterable.
```

And in fact we can tell Inform that the player starts on the pedestal with this line:

```
The player is on a pedestal.
```

Now the player will begin there rather than just in the Center Ring.

This last bit is an entirely unnecessary bit of local color, but if we're going to keep getting into and out of the lion's cage, we ought to expect him to take notice:

```
Every turn when the player is in the cage:
    if a random chance of 1 in 2 succeeds, say "The lion
eyes you with obvious discontent.";
    otherwise say "Though the lion does not move, you
are aware that it is watching you closely."
```

Randomness is explained more completely in the chapter on Change, and every turn rules in the chapter on Time.

Finally, we might want a container whose interior is modeled as its own separate room: say, a magician's booth in which volunteers are made to disappear.

```
The magician's booth is a container in Center Ring. "Off
to one side is a magician's booth, used in disappearing
acts. The exterior is covered with painted gilt stars." The
```

```
booth is enterable, open, not openable, and fixed in
place.
```

Now we create our other location:

```
Inside from the Center Ring is the Starry Vastness.
```

...which handles the case of the player typing >IN. (We will not assume by default that he wants to get into the cage with the lion, this being obviously perilous.) But we also want to make sure that the player who types >ENTER BOOTH winds up in the same place, so we should add:

```
Instead of entering the magician's booth: try going
inside.
```

```
Test me with "get in cage / open cage / get in cage / z /
close cage / out / open cage / get on pedestal / get off /
look / enter booth / out".
```

Example **Princess and the Pea** *WI*

The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

The main point here is that we need to figure out where the stack meets the floor:

```
"Princess and the Pea"
```

```
The Topmost Turret is a room. A mattress is a kind of
supporter. A mattress is always enterable. A mattress is
portable.
```

A large mattress is a mattress in the Turret. A medium mattress is a mattress in the Turret. A small mattress is a mattress in the Turret.

Instead of sleeping when the player is on a mattress (called the bed):
    let the item be the bed;
    while the holder of the item is not a room:
        let the item be the holder of the item;
    say "You can still feel something very uncomfortable under [the item]."

Instead of sleeping:
    say "You can't sleep standing up!"

Instead of looking under a mattress, say "You scout around, but are unable to determine what's causing you this discomfort. If only your maid Winnie were here. She's very good at this."

Test me with "sleep / enter small / sleep / get up / get small / put small on medium / get on small / sleep / get up / g / get medium / put medium on large / get on small / look / sleep".

---

361 **ExampleKiwi** *WI*

Creating a raised supporter kind whose contents the player can't see or take from the ground.

Suppose we want there to be some high shelves in our game, which the player can't get at unless he's standing on a prop of some kind. (This is a pretty hoary and over-used puzzle, but there may still be occasions when it becomes useful again.)

In order to resolve this, we want to set up a raised supporter kind. When something is on a raised supporter, it should be mentioned to the player

only if the player is in the right position (i.e., standing on something) and otherwise omitted from the description entirely.

```
"Kiwi"

Section 1 - Procedure

A raised supporter is a kind of supporter.

For printing a locale paragraph about a raised supporter
(called the high place):
    if the player is on a supporter (called the riser):
        say "Up on [the high place] (and only visible
because you're on [the riser]) [is-are a list of things on
the high place].";
    otherwise:
        say "The [high place] is above you."
```

Note that here we don't continue the activity because we want to completely replace the normal behavior of describing what is on supporters.

```
Definition: a thing (called target item) is out of reach:
    if the player is on a supporter, no;
    if the target item is on a raised supporter, yes;
    no.
```

Now we also need to prevent the player from interacting with things that are out of reach:

```
Before doing something:
    if the noun is out of reach or the second noun is out of
reach:
        say "You can't reach from down here." instead.
```

...or restoring things to the shelves while the player is in the wrong

position...

```
Instead of putting something on a raised supporter when
the player is not on a supporter:
    say "You can't reach from down here."
```

And raised supporters shouldn't be searchable from the ground either:

```
Instead of searching or examining a raised supporter
when the player is not on a supporter:
    say "You can't see from down here."
```

Finally, we need to tackle the case where the player types GET ALL FROM SHELF, because we don't want to list the objects up there if the player can't even see them. We use a rule for deciding whether all includes in order to tell Inform not to consider items that can't be reached, and then we adjust the parser error so that it's a little more instructive than "There are none at all available!", which is what the response would otherwise be:

```
Disallowed-all is a truth state that varies. Disallowed-all
is false.


Rule for deciding whether all includes an out of reach
thing:
    now disallowed-all is true;
    it does not.


Rule for printing a parser error when the latest parser
error is the nothing to do error and the player is not on a
supporter:
    if disallowed-all is true:
        say "Whatever might be up there, you can't see or
reach from down here.";
    otherwise:
        make no decision.
```

A first action-processing rule:
    now disallowed-all is false.

Section 2 - Scenario

The Bottom of the Nursery is a room. "Ever since you ate that mysterious cake, you've been even shorter than usual."

The high shelf is a raised supporter in the Nursery. It is scenery. On the high shelf are a kiwi-green ball and a stuffed dodo.

The step-ladder is an enterable supporter in the Nursery. Understand "ladder" as the step-ladder.

Test me with "x shelf / search shelf / get dodo / get all from shelf / stand on ladder / get all from shelf / search shelf / get off / put all on shelf / get all from shelf / stand on ladder / put all on shelf".

Example **Circle of Misery**

WI

Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

The only point we need to be careful about is that the carousel is simulated twice over, in the following text: once in the built-in way that objects are inside other objects, so that the luggage items are objects contained in the carousel object; but then again by the "circle of misery" list, a ring buffer keeping track of what order things are in. We need to be careful that these two records do not fall out of synchrony: anything put into the carousel must be added to the list, anything taken out must be removed. (In fact we forbid things to be added, for simplicity's sake.)

"Circle Of Misery"

Luggage item is a kind of thing. The blue golf bag, the leopardskin suitcase, the green rucksack and the Lufthansa shoulder-bag are luggage items.

Heathrow Baggage Claim is a room. The carousel is a container in Heathrow. "The luggage carousel, a scaly rubbered ring, does for the roundabout what Heathrow Airport does for the dream of flight: that is, turns the purest magic into the essence of boredom, only with extra stress. [if the number of entries in the circle of misery is 0]For once it stands idle. Perhaps it's broken. [otherwise]The baggage approaching you now: [the circle of misery with indefinite articles]."

The circle of misery is a list of objects that varies.

When play begins:
    now all the luggage items are in the carousel;
    add the list of luggage items to the circle of misery.

The list "circle of misery" is our ring, in which entry 1 is considered to be the position of whichever bag is currently frontmost. And here it goes, round and round:

Every turn when the number of entries in the circle of misery is not 0:
    rotate the circle of misery;
    let the bag be entry 1 of the circle of misery;
    say "The carousel trundles on, bringing [a bag] to within reach."

After taking a luggage item (called the bag):

remove the bag from the circle of misery, if present;
say "Taken."

Before doing something with a luggage item (called the bag) which is in the carousel:
if the bag is not entry 1 of the circle of misery, say "[The bag] is maddeningly out of range. You'll have to wait for it to come round." instead.

Instead of inserting something into the carousel, say "In recent years, the authorities have tended to frown upon depositing bags in random places at airports."

Test me with "get suitcase / get suitcase / get suitcase / get suitcase / look / get golf bag / look / get golf bag".

Example **Yolk of Gold**

Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

Suppose that for dramatic effect we would like the player to find the thing he is looking for always in the last drawer he opens...

"Yolk of Gold"

The Turret is a room. "A cramped little room distinguished chiefly by the spiral staircase that descends from it. The windows look out over the rooftop."

The Rooftop is outside from the Turret.

The staircase is an open unopenable door. It is scenery. It is below the Turret and above the Library. The description is "A winding stair carved out of the single trunk of a massive tree, all in a dark wood; the outside of the stairs left unfinished with the bark still on, but the treads worn smooth by long and constant use."

The description of the Library is "Here, it seems, you have found your mark: books line both walls, a dark carpet lies on the floor, and a strange dress hangs up in a presentation case. And the thing you were told to look for, a desk with three drawers.

A spiral staircase leads up and out."

The cherry desk is scenery in the library. The description is "A deep, satin-lustrous cherry, with scrollwork legs and gilt touches. The years have not been kind, and it has cracked and split in several places; the finish is damaged, and where there is inlaid mother of pearl, it is beginning to come up from its bed. But it is still a sound piece, and features three drawers."

After examining the desk for the first time, say "(Your employers were able to tell you to look for it, but not which drawer to look in. Typical.)"

A drawer is a kind of container. A drawer is always openable and closed. The description of a drawer is "The usual drawer of heavy wood, inadequately smoothed for ease of use."

The top drawer is a drawer. The middle drawer is a drawer. The bottom drawer is a drawer. The top drawer,

the middle drawer, and the bottom drawer are part of the desk. A drawer can be explored or unexplored. A drawer is usually unexplored. Instead of searching a closed drawer, try opening the noun.

After opening a drawer when no drawers are explored:
    now the noun is explored;
    say "There is a tremendous screech, but nothing whatsoever inside, not even dust."

After opening an unexplored drawer when exactly one drawer is explored:
    now the noun is explored;
    say "You struggle to open this one a bit more quietly, conscious all the time of noise... but no, it's empty. Just stands to reason."

After closing a drawer:
    say "There is a racket of wood protesting against wood as you do so, which makes you wonder if you hadn't better just leave them open from now on. The unhappy owner is probably going to catch on soon enough anyway."

Before opening an unexplored drawer when exactly two drawers are explored:
    move the mysterious thing to the noun;
    now the noun is explored.

There is a mysterious thing. The description is "A very familiar-looking hemispherical weight of metal, goldish in tone though perhaps not made of gold. This one has a slight stickiness about the bottom surface."

Instead of opening the desk when at least one drawer is unexplored:

```
    let the next drawer be a random unexplored drawer;
    say "(starting with [the next drawer])[line break]";
    try opening the next drawer.


Instead of looking under the desk when no drawers are
explored:
    say "You carefully survey the ground around the
desk. There don't seem to be any hidden tripwires or
traps to prevent you from having a look in the drawers."


Instead of looking under the desk when at least one
drawer is unexplored:
    say "Nothing there either. Thoroughness is a virtue
with tedium as a side effect, as your mother used to say
-- but they always counted her a trifle slapdash."


Instead of looking under the desk when all drawers are
explored:
    say "There's nothing down there, but this doesn't
come as a great surprise."


Instead of searching the desk when at least one drawer
is unexplored:
    move the mysterious thing to the player;
    say "You perform a hasty, squeaky search of ";
    if no drawer is explored, say "all three drawers,
discovering nothing and nothing in rapid succession. But
on the third drawer you ";
    if exactly two drawers are unexplored, say "the
remaining two drawers. There's nothing in the second,
but in the third you ";
    if exactly one drawer is unexplored, say "the last
drawer. In it, you ";
    say "turn up a promising hemispherical object.";
    now all the drawers are explored.
```

Perhaps, just for fun, we have all the other scenery draw the player's attention back to the main point, as well.

The carpet is scenery in the Library. The description is "It is too dark for you to make out details of the antique design, which seems dimly to represent an early voyage to the moon, with ships and the travelling stars." Understand "rug" as carpet.

Instead of looking under the carpet:
    if none of the drawers are explored, say "You peel up a corner of the rug gingerly; nothing results. A perfectly ordinary rug, then." instead;
    if all of the drawers are explored, say "No time for that kind of nonsense. You'd better get out and away while you can." instead;
    if some of the drawers are explored, say "No time for that nonsense. The desk's what you want now; what you came for won't be embedded in the flooring." instead.

The glass presentation case is transparent scenery in the Library. The description is "The case is taller than you are, framed in wood, with large panels of glass, the better to display the remarkable contents."

Instead of attacking the glass presentation case:
    say "The glass pane of the presentation case shatters, throwing fine glass everywhere, including over the delicate museum-piece inside. Nor does the noise pass unnoted: only a second passes before there are footsteps in the hall, and though you make for the concealing darkness and escape of the turret, you are not swift enough. The servants are soon on you, and you are made to regret, quite painfully, this casual act of vandalism.";
    end the story saying "You have lost your opportunity."

The strange dress is a wearable thing in the glass case. The description is "Not the sort of dress that anyone wears now: such elaboration would be ludicrous. It drips gold -- gilt lace, ruffles of trimmed gold, shimmering golden tracery -- dulled here and there by the sinister black of faceted jet."

Test me with "d / x case / x dress / x carpet / look under carpet / look under desk / x desk / open top drawer / close top drawer / look under desk / open bottom drawer / close bottom drawer / open middle drawer / get thing / look under carpet / look under desk / up".

---

397 **Example** **Swigmore U.** *WI*

Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

Inform's default assumption is that if a player on an enterable object drops something, the dropped article winds up beside him on the same supporter or in the same container. This makes lots of sense for a dais, say, or a king-sized bed. It's a little less sensible if the enterable supporter in question is a bar stool or the like. So suppose we want to add a new kind of supporter called a perch, where everything dropped lands on the floor.

There are actually several ways of implementing this, but one of them is to reach right into the drop action and replace the "standard dropping rule" with a different one of our own invention -- like this:

"Swigmore U."

Moe's Tavern is a room. The bar is an enterable supporter in Moe's. A drink is a kind of thing. On the bar is a drink called a flaming Homer.

A perch is a kind of supporter. A perch is always enterable. The stool is a perch in Moe's.

The player carries a dead field mouse and a tomacco fruit.

```
The sophisticated dropping rule is listed instead of the
standard dropping rule in the carry out dropping
rulebook.


This is the sophisticated dropping rule:
    if the player is on a perch (called the awkward
position):
        let place be the holder of the awkward position;
        move the noun to the place;
    otherwise:
        move the noun to the holder of the player.


Test me with "sit on stool / drop mouse / look / get up /
look".
```

Now the carry-out behavior of the dropping action has been changed, but we haven't had to interfere in the checks or reporting at all. The rest of the action works just as it always did.

Of course, maybe we do want to change the way the action is reported, to make it clearer to the player where the dropped article wound up:

```
The sophisticated report dropping rule is listed instead
of the standard report dropping rule in the report
dropping rulebook.


This is the sophisticated report dropping rule:
    say "You drop [the noun] on [if the holder of the noun
is a room]the ground[otherwise][the holder of the noun]
[end if]."
```

ExampleU-Stor-It

A "chest" kind which consists of a container which has a lid as a supporter.

Suppose we want to write a game in which there are a number of chests. Each of these chests will be a container, but have a lid which is a supporter.

```
"U-Stor-It"

Section 1 - Assemblies and Supporters

A chest is a kind of container. A chest is always
openable. A chest is usually fixed in place. A chest is
usually closed. The specification of a chest is
"Represents a container with a separately implemented
lid; the lid is itself a supporter."

A lid is a kind of supporter. A lid is part of every chest.
The specification of a lid is "A supporter attached to a
chest, which can only support things when the chest is
closed."
```

(The "specification" of a kind is not really a property, and is used instead to describe the kind in the Index. So the text of these specifications is never found in the game.) Of course, this doesn't get us very far. We will also want the game to correctly interpret variations on "open the chest" and "close the lid", redirecting actions appropriately.

```
Section 2 - Opening and closing

Before opening a lid which is part of a chest (called the
item):
    try opening the item instead.

Before closing a lid which is part of a chest (called the
item):
    try closing the item instead.
```

```
Before opening a chest (called the box) when something
is on a lid (called the obstruction) which is part of the
box:
    repeat with item running through things on the
obstruction:
        say "(first removing [the item])";
        try taking the item.


Instead of opening a chest when something is on a lid
(called the item) which is part of the noun:
    say "You'd have to remove [the list of things on the
item] from the lid first." instead.


Instead of looking under a lid which is part of a chest
(called the item):
    try opening the item.
```

We may also want to be able to deal with "put in" and "put on" appropriately, even if the player names the wrong part of the object:

```
Section 3 - Insertion and Support


Before inserting something into a lid which is part of a
chest (called the item):
    try inserting the noun into the item instead.


Before putting something on a chest when a lid (called
the item) is part of the second noun:
    try putting the noun on the item instead.
```

Furthermore, we don't want the player to be able to put things on the lid while the chest is open:

```
Before putting something on a lid which is part of an
```

```
open chest (called the item):
    say "(first closing [the item])";
    try closing the item.


Instead of putting something on a lid which is part of an
open chest (called the item):
    say "[The item] would need to be closed first."
```

And then we may also want a couple of rules for describing our assembled object nicely:

```
Section 4 - Description in Rooms


Instead of examining a closed chest when something is
on a lid (called the top) which is part of the noun:
    say "[The noun] is closed, and there [is-are a list of
things on the top] on top."


After printing the name of a chest (called the item) while
listing contents of a room:
    if a lid (called the second item) which supports
something is part of the item:
        say " (on which [is-are a list of things on the
second item])";
        omit contents in listing.
```

Now we are free to create entire treasure rooms at a single blow:

```
Section 5 - U-Stor-It Facility


The U-Stor-It Facility is a room. The sea trunk, the
shipping crate, and a metal box are chests in the U-
Stor-It Facility. The metal box contains a sapphire, a
gold coin, and a signed photograph of Babe Ruth.
```

Even though we have never explicitly defined it, the metal box has a

"metal box's lid", which we can use at need.

```
The metal box's lid supports a small card. The
description of the small card is "It reads, 'Back in 5 mins
- Pandora.'"


Test me with "open trunk / x card / open metal box / put
all in metal box / get card / put card on box".
```

ExampleVersailles

A mirror which will reflect some random object in the room.

One of the advantages of descriptions is that we can use them to pick an item randomly from a specified category. (For more on this possibility, see the Change chapter sections on randomness.)

For instance, suppose we wanted to create a mirror in which the player would see some item from the room reflected. We might write

```
Instead of searching the mirror:
    say "You see [a random thing in the location]
reflected back at you."
```

This is the same as "a random thing which is in the location": phrase "in..." can be used briefly in Inform as it can in English.

But, on a little more thought, we might want to expand on this: the mirror perhaps should reflect not only things that are in the room, but anything that the player can see (even if it's on a supporter or carried by someone). So then we might instead write

```
Instead of searching the mirror:
    say "You see [a random visible thing] reflected back
at you."
```

There's still a risk, though, that this will produce the response

```
You see the mirror reflected back at you.
```

because, of course, the mirror is itself visible. So instead we might write

```
Instead of searching the mirror:
    say "You see [a random visible thing which is not the
mirror] reflected back at you."
```

```
"Versailles"
```

```
The Hall of Plywood Boards is a room. "The Hall of
Mirrors is under reconstruction: it is currently a dank
tunnel enlivened only by short placards about the history
of the room.
```

```
As though to mock tourists such as yourself who bought
their tickets without knowing this, the officials have left
uncovered a single panel of mirror."
```

```
The mirror is scenery in the Hall of Plywood Boards.
Understand "panel" or "panel of mirror" as the mirror.
The description is "Lovingly restored to shimmering
brilliance, it suggests how marvelous this room would be
if you had had the good sense to arrive after the
renovations were complete."
```

```
Some tourists are a person in the Hall of Plywood
Boards. The tourists are scenery.
```

```
Instead of searching the mirror:
    say "You see [a random visible thing which is not the
mirror] reflected back at you."
```

```
Test me with "x mirror / look in mirror / g".
```

A final note: we use "searching" here because Inform understands both SEARCH THING and LOOK IN THING as the searching action, and the player is most likely to type LOOK IN MIRROR in order to see the reflection there. In the absence of an example, we can discover the relationship between actions and their command vocabulary in one of two ways. A complete list of actions and the vocabulary associated with them is available in the Actions index. Alternatively, we can type ACTIONS at a prompt, followed by LOOK IN MIRROR, and get the response

```
[searching the mirror]
You find nothing of interest.
[searching the mirror - failed the can't search unless
container or supporter rule]
```

...which tells us that Inform is understanding the action as "searching the mirror".

---

449 **Example** **Modern Conveniences** *WI*

Exemplifying the kind of source we might use in writing extensions for kitchen and bathroom appliances.

Suppose we want to write an extension or other portable code which defines a "kitchen" kind of room and a "bathroom" kind of room. All kitchen rooms we create in the future will automatically contain standard kitchen appliances: fridge, freezer, sink with taps, counters, cabinets, and a stovetop with built-in oven. Similarly, all bathrooms will have sinks, baths, cabinets, and toilets, and respond to some standard interactions.

We would do this with a standard assembly:

```
"Modern Conveniences"

Section 1 - Kitchens
```

```
A kitchen is a kind of room.
```

```
A refrigerator is a kind of container. A refrigerator is
usually closed and openable. A refrigerator is usually
fixed in place. A refrigerator is usually scenery.
Understand "fridge" as a refrigerator.
```

```
A freezer compartment is a kind of container. A freezer
compartment is usually closed and openable. A freezer
compartment is part of every refrigerator.
```

Now: we're going to want many of the items in our kitchen to have switches, and to handle input sensibly regardless of whether the player types TURN ON STOVE or TURN ON STOVE SWITCH. (This is apparently a stove with only one burner.) For convenience, we'll define an "includes" relation:

```
Inclusion relates a thing (called X) to a thing (called Y)
when Y is part of X. The verb to include means the
inclusion relation.
```

```
A stove is a kind of supporter. It is usually scenery.
An oven is a kind of container. An oven is usually
openable and closed. One oven is a part of every stove.
A switch is a kind of device. A switch is part of every
stove. A switch is part of every oven.
Understand "[something related by reversed
incorporation] switch" as a switch.
```

What follows is fairly straightforward, but notice that we are somewhat obsessively naming every rule. This is much more important in extensions (where someone else may need to manipulate our code from within their own source) than it is when we are simply composing source text for ourselves.

```
Setting action variables for opening a stove (this is the
```

stove-opening rule):
    let relevant oven be a random oven which is part of the noun;
    now the noun is the relevant oven.

Setting action variables for switching on something which includes a switch (this is the redirecting switches for switching on rule):
    let relevant switch be a random switch which is part of the noun;
    now the noun is the relevant switch.

Setting action variables for switching off something which includes a switch (this is the redirecting switches for switching off rule):
    let relevant switch be a random switch which is part of the noun;
    now the noun is the relevant switch.

Before printing the name of a switch (called target) (this is the switch identification rule):
    say "[random thing which includes the target] ".

A sink is a kind of container. A sink is usually fixed in place and scenery. A tap is a kind of switch. A tap is part of every sink. Understand "faucet" or "taps" as a tap. Understand "[something related by reversed incorporation] tap/faucet/taps" as a tap.

Instead of opening a tap, try switching on the noun.
Instead of closing a tap, try switching off the noun.

Report switching on a tap (this is the standard report switching taps on rule):
    say "You turn on [the noun]." instead. [since "switch on" sounds weird in this context.]

```
Report switching off a tap (this is the standard report
switching taps off rule):
    say "You turn off [the noun]." instead.
```

```
After examining something which includes a switched
on tap (called relevant tap) (this is the report flowing
water rule):
    say "The water is flowing from [the relevant tap]."
```

```
A drain is a kind of container. A drain is part of every
sink. Understand "plughole" as the drain. Understand
"[something related by reversed incorporation] drain/
plughole" as a drain.
```

```
Instead of inserting something into a drain (this is the no
clogging drains rule), say "Pointless."
```

This is probably about as far as we want to go in a generic simulation: it is tempting to code up water, drains down which the player can lose objects, sinks that get clogged and overflow, and so on; but the more we embellish in these ways, the more likely the end result would be disruptive to individual games. For right now what we're aiming for is something simple which provides the basic interactions a player might expect in this kind of room, but which does not have any significant implications for the surrounding world model.

A particularly conservative author might even want to make it turn out that the water has been shut off and nothing flows from the taps: in the extension documentation, we might want to include a line or two of example showing how this might be done by changing or removing the relevant rules of our extension.

```
A cabinet is a kind of container. A cabinet is usually
openable and closed. It is scenery.
    Understand "cupboard" or "cupboards" or "cabinets"
as a cabinet.
```

```
A counter is a kind of supporter. It is scenery.
    Understand "countertop" as a counter.


A cabinet is in every kitchen.
A counter is in every kitchen.
A refrigerator is in every kitchen.
A sink is in every kitchen.
A stove is in every kitchen.


Section 2 - Bathrooms


A bathroom is a kind of room.


A toilet is a kind of supporter. Understand "john" as a
toilet. A toilet is usually fixed in place and enterable and
scenery.


A bath is a kind of container. A bath is usually a fixed in
place and enterable and scenery. A tap is part of every
bath. A drain is part of every bath. Understand "bathtub"
or "shower" as a bath.


A sink is in every bathroom.
A toilet is in every bathroom.
A bath is in every bathroom.
A cabinet is in every bathroom.
```

If we were feeling particularly ambitious and inclined toward interior decoration, we could add bath mats, mirrors, plungers, toilet brushes, overhead lighting, towel racks, scented candles, boxes of facial tissue, shampoo bottles, scrubbing loofahs, etc. ad nauseam; but we'll keep it relatively simple for now. Of course, if we have a toilet, we pretty much have to accept that the player will try to make use of it:

```
Understand "flush [toilet]" or "use [toilet]" as a mistake
("You have no need at the moment.").


Understand "take shower" or "take bath" or "bathe" or
"wash" as bathing. Bathing is an action applying to
nothing.


Check bathing (this is the restrict baths to bathrooms
rule):
    if the location is not a bathroom, say "This isn't the
place." instead.


Check bathing (this is the block bathing rule):
    say "You haven't time for a bath." instead.


Washing is an action applying to one thing. Understand
"clean [something]" or "wet [something]" or "wash
[something]" as washing.


Instead of washing the player, try bathing.


Check washing (this is the restrict washing to the
proximity of sinks rule):
    unless the player can touch a sink, say "This isn't the
place." instead.


Check washing (this is the block washing rule):
    say "It doesn't seem worth the bother." instead.
```

Now we might put this to work in a short example.

One slight challenge lies in giving these assembled pieces separate descriptions. When we have an assembly that adds parts to objects, we can then talk about (for instance) "the stove's switch" elsewhere in the

code. But items that have been assigned rooms are not named in the same way, so we cannot talk about "the Industrial Kitchen's stove" in our code as a way to assign it a description or special behavior. In quite a simple example, we could make the descriptions of the kind simply be the descriptions we want for the individual items:

Section 3 - An Example We Might Offer

Our Household Kitchen is a kitchen.

The Tiny Bathroom is a bathroom. It is west of Our Household Kitchen.

The description of a stove is "Scrupulously polished."

The description of a refrigerator is "It is baby blue and has the contours of a 50[']s chevy. One of these days it really will break down, but it's been serving your family faithfully since your grandmother's honeymoon."

Test me with "x refrigerator / open fridge / x freezer / look in freezer / open freezer / turn on stove / turn on oven / x oven switch / turn off oven switch / turn off stove switch / turn on taps / x sink / w / x sink / turn on sink / take bath / use toilet".

In a game that featured multiple bathrooms and kitchens, this wouldn't be enough; our author might give the stove kind (say) a description that checked its location, as in

The description of a stove is "[if in Industrial Kitchen]A massive stainless steel stove-top with six burners[otherwise]Your standard four-burner item[end if]."

or create an

```
Instead of examining the stove in the Industrial Kitchen:
...
```

sort of rule for those objects he wanted to describe specially; or he might use a when play begins rule to initialize a few things:

```
When play begins:
    let N be a random stove in the Industrial Kitchen;
    move the boiling pot to N;
    change the description of N to...
```

Or we might even (if we anticipate lots of these kinds of amendments) want to rig up something more complex that finds the descriptions of appliances in a table, rather than relying on their individual description properties. This can all be done, but it is only interesting as long as it remains genuinely labor-saving: that is, as long as the convenience of having the assembly is greater than the annoyance of writing special rules to cover for the automation. In the end, the "kitchen" and "bathroom" room types are likely to be most useful for authors who want to include the standard props but not actually make them a critical part of the game; if stoves and sinks have more of a starring role in the production, authors may be better off coding them or at the very least placing them by hand, as in

```
The Industrial Kitchen is a room.
```

```
Thor is a stove in the Industrial Kitchen. It supports a
boiling pot.
```

All these quirks are things that we (as the extension author) want to think out in advance: we should ideally warn authors about possible pitfalls in using our extension (if we can think of them) and point out ways of customizing the behavior (if there are interesting ways).

Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

9.6. Reading Matter; *§9.7. Painting and Labeling Devices*; *§9.8. Simple Machines*; *§9.9. Televisions an*

Contents of *The Inform Recipe Book*

Chapter 8: Vehicles, Animals and Furniture

Chapter 10: Physics: Substances, Ropes, Energy and Weight

Indexes of the examples

§9.1. Food

Inform provides an either/or property called "edible" and an action, "eating", for consuming edible things:

```
The lardy cake is edible. After eating the lardy cake, say "Sticky but
delicious."
```

For eating something not immediately to hand, see **Lollipop Guild**. **Delicious, Delicious Rocks**, conversely, adds a sanity check which prevents the player from automatically taking inedible things only to be told they can't be eaten.

Inform does not normally simulate taste or digestion, but to provide foods with a range of flavours, see **Would you...?**; to make eating different foods affect the player differently, see **Stone**, or for the extreme case of poisoning foods, **Candy**. In **MRE**, hunger causes the player problems unless he regularly finds and eats food.

See **Liquids** *for things to drink*

See **Dispensers and Supplies of Small Objects** *for a pizza buffet table from which the player may take all the slices he wants*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to Chapter 8: Vehicles, Animals and Furniture: §8.5. Kitchen and Bathroom

Onward to §9.2. Bags, Bottles, Boxes and Safes

Example 48: ★ **Would you...?** Adding new properties to objects, and checking for their presence.

Example 133: ★ **Candy** One of several identical candies chosen at the start of play to be poisonous.

Example 143: ★ **MRE** Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

Example 391: ★ **Stone** A soup to which the player can add ingredients, which will have different effects when the player eats.

Example 196: ★★★ **Delicious, Delicious Rocks** Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

---

§9.2. Bags, Bottles, Boxes and Safes

The kind "container" allows one thing to contain others. Things are sometimes containers automatically, sometimes by instruction:

The match is in the matchbox. The bucket is a container.

The matchbox, like the bucket, is a container. Containers come in all sizes and have a variety of behaviours, mainly controlled by the properties we give them: they can be "open" or "closed", "opaque" or "transparent" (when closed), "openable" or not, "lockable" or not, "enterable" or not. The basic ideas of containment are to do with carrying and sometimes hiding the contents, and Inform makes this easy. Allowing for locking and unlocking is again straightforward:

```
The strongbox is a locked container. The little steel key unlocks the
strongbox.
```

Two built-in extensions enhance and modify the behavior of locks and keys: Locksmith automates a number of steps, automatically unlocking doors when trying to open them (for instance). Skeleton Keys allows us to define multiple keys that unlock the same object, rather than being restricted to one matching key per item.

For a container with a combination lock, rather than a key, see **Safety**; for a more sophisticated safe requiring digits dialed over multiple turns, see **Eyes, Fingers, Toes**.

Trachypachidae Maturin 1803 provides a bottle that is stoppered with a cork: when it is closed, the cork is part of the bottle, but otherwise the cork becomes a separate object we can carry around.

The normal assumption is that there is no problem with any two portable items being carried together, but in reality they may affect each other. (For effects like magnetism, or getting each other wet, or setting each other on fire, see the Physics chapter.) Here is a cat which, if boxed up with one or more items of food, will eat something each turn until all is gone:

```
The player carries a wicker basket and a scarlet fish. The cat is an
animal in the wicker basket. The fish is edible.


Every turn when the cat is in a container (called the bag) and
something edible (called the foodstuff) is in the bag:
    remove the foodstuff from play;
    say "With mingled sounds of mewing and chomping, the cat
nibbles up [the foodstuff]."
```

The examples below provide subtler effects, adapting text to the current situation. In **Cinco**, the container's name changes depending on what it contains: putting beef in a taco allows the player to call it a SHREDDED BEEF TACO. In **Unpeeled** and **Shipping Trunk**, the description of something inside a container changes according to other things are alongside it. This is taken further in **Hudsucker Industries**, which

describes the contents of a container as a group.

Finally, any action that destroys a container has to consider what to do with the things inside. **Fallout Enclosure** demonstrates a zapping action that destroys cash registers and shelves but leaves their contents tidily behind.

See **Liquids** *for a SHAKE command that makes containers rattle when there are contents*

See **Glass and Other Damage-Prone Substances** *for opening containers by cutting into them*

See **Fire** *for fire damage that spreads between containers and their contents, leaving fireproof objects intact*

See **Volume, Height, Weight** *for containers breaking under the weight of their contents*

See **Heat** *for keeping things warm in insulated containers*

See **Furniture** *for chests with lids that can support other objects*

See **Modifying Existing Commands** *for ways to allow the player to unlock with a key he isn't currently holding*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.1. Food

Onward to §9.3. Clothing

Example 298: **Safety** A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.

Example 319: **Cinco** A taco shell that can be referred to (when it contains things) in terms of its contents.

Example 339: **Shipping Trunk** A box of baking soda

whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

Example 344: ★ **Unpeeled** Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

Example 433: ★ **Eyes, Fingers, Toes** A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

Example 340: ★★ **Trachypachidae Maturin 1803** Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

Example 342: ★★ **Hudsucker Industries** Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

Example 35: ★★★ **Fallout Enclosure** Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

§9.3. Clothing

A person can wear any (portable) thing which has the "wearable" property. (This property seldom needs to be quoted because it is deduced automatically from sentences like "Trevor wears a red hat.")

In most traditional IF, clothing is only used when it is exceptional in some way. That is, we ignore the three to eight different garments most people are wearing at any given time - the everyday clothes which people wear without thinking about them - and only simulate the unexpected extras: a borrowed jaunty red hat, a radiation-proof space suit, and so on.

These unusual garments turn up only occasionally in play and usually one at a time, so Inform does not normally provide rules to restrict how much or little is worn, or in what unlikely combinations. **Get Me to the Church on Time** categorises clothing by body area (trousers for lower body, shirts for upper); **Bogart** by layer, distinguishing underwear from outer garments. **What Not To Wear** combines both into a general-purpose system adequate for most kinds of clothing situations.

See **Kitchen and Bathroom** *for a simple mirror implementation, which could be adapted to reflect what the player is currently wearing*

Hays Code is a somewhat stripped down version.

Clothes are normally single things which have no function other than display and concealment, but **Being Prepared** gives them pockets which act as containers, and **Some Assembly Required** allows clothes to be stitched together from pieces of cloth.

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.2. Bags, Bottles, Boxes and Safes

Onward to §9.4. Money

Example 56: **Being Prepared** A kind for jackets, which always includes a container called a pocket.

Example 332: **Some Assembly Required** Building different styles of shirt from

component sleeves and collars.

Example 338: ★ **Hays Code** Clark Gable in a pin-striped suit and a pink thong.

Example 242: ★★ **What Not To Wear** A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

Example 46: ★★★ **Get Me to the Church on Time** Using kinds of clothing to prevent the player from wearing several pairs of trousers at the same time.

Example 234: ★★★ **Bogart** Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

---

§9.4. Money

Money could be anything which the two people in a bargain both agree is valuable. Here, the player and an ogre agree on a copper coin as money:

```
The player carries a copper coin. The ogre carries a rock cake. The
cake is edible.



Instead of giving the coin to the ogre:
    now the ogre carries the coin;
    now the player carries the cake;
    say "The ogre grunts and hands you a rock cake."
```

Now Inform does provide an action, "buying", and a command for it, BUY, but they ordinarily respond simply "Nothing is on sale." This is no longer true, so we should

make BUY CAKE work. The difficulty here is that a command like BUY CAKE does not specify what should be handed over in exchange. Here we just check that the player has the coin, but in principle we could check for any of a range of monetary tokens - coins, notes, cheque book, debit card, and so on.

```
Instead of buying the cake:
    if the player has the coin, try giving the coin to the ogre;
    otherwise say "You have no money."
```

In more advanced economies, where shopping replaces barter, the seller will stock a wide range of differently priced goods. For a tabulated catalogue of wares, see **Introduction to Juggling**: to allow the player to negotiate prices, see **Money for Nothing**. In both of those examples, the player's current financial worth is simulated only as a current total amount of money carried - say, $2.50. This is typical, because in most situations what matters is how much money is in the pocket, not how it is made up. Money behaves more like a liquid than a set of items: hence terms like "liquidity", "cash flow" or **Frozen Assets** - the name of the simplest example demonstrating this. If we really need a comprehensive simulation down to pieces of currency - where it makes a difference carrying four quarters rather than a dollar bill, because the quarters can be fed into a vending machine - see **Nickel and Dimed**.

Fabrication takes the problem in a different direction, making calculations about the cost of a new garment based on the price of the pattern, the quantity of fabric required, and the value of the fabric type chosen -- showing off what we can do with unit multiplication in Inform.

Widget Enterprises explores the challenge of pricing widgets for maximum profit, given certain necessary costs and customers with varying willingness to pay.

See **Actions on Multiple Objects** *for an implementation of giving that allows the player to offer multiple objects at once, where their combined value determines whether they are accepted*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.3. Clothing

Onward to §9.5. Dice and Playing Cards

Example 263: **Widget Enterprises** Allowing the player to set a price for a widget on sale, then determining the resulting

sales based on consumer demand, and the resulting profit and loss.

Example 264: ★ **Frozen Assets** A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping.

Example 265: ★★ **Money for Nothing** An OFFER price FOR command, allowing the player to bargain with a flexible seller.

Example 269: ★★ **Fabrication** A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

Example 262: ★★★ **Nickel and Dimed** A more intricate system of money, this time keeping track of the individual denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change.

Example 283: ★★★ **Introduction to Juggling** Assortment of equipment defined with price and description, in a table.

---

§9.5. Dice and Playing Cards

Most toys are single things, and no harder to create than any other small items, but games often require a multitude of tokens to be combined, and this can be logistically tricky.

The classic example is a pack of playing cards, where the player must individually control 52 items but without fussy commands or verbose text being printed back. **Jokers Wild** provides a simple "one card at a time" approach; **Tilt 1** is more sophisticated, with 52 independently accessible cards; **Tilt 2** can further judge the value of a selection of cards - the ranking of a poker hand.

Drawing cards from a shuffled pack is only one source of randomness. Games of chance also involve items drawn from a bag: **Wonka's Revenge** provides just such a lottery. More often, dice are thrown. A single die is easy enough:

> The die is carried by the player. After dropping the die: say "It lands with [a random number from 1 to 6] uppermost." Understand "roll [something]" as dropping.

Quick, but not very good. Most dice games involve rolling more than one die at a time, to get a more interesting distribution of outcomes: they may also involve special rules applying to doubles, for instance. See **Do Pass Go**.

See **Typography** *for on-screen notations for chess and card games*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.4. Money

Onward to §9.6. Reading Matter

Example 129: **Do Pass Go** A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

Example 175: **Wonka's Revenge** A lottery drum which redistributes the tickets inside whenever the player spins it.

Example 275: **Jokers Wild** A deck of cards which can be shuffled and dealt from.

Example 318: **Tilt 1** A deck of cards with fully

implemented individual cards, which can be separately drawn and discarded, and referred to by name.

Example 407: ★★★ **Tilt 2** A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

---

§9.6. Reading Matter

Many things can be read, from warning notices to encyclopaedias, and a range of techniques is needed to provide them because the quantity of text, and how it presents itself, can vary so much. With a small amount of very large type, the player should not need any command at all to read the text:

> The road sign is in the Junction. The road sign is fixed in place. "A road sign points north: 'Weston on the Green - 6'."

If the print is smaller, or the object portable, the player will expect to use the EXAMINE command:

> The business card is in the Junction. The description is "'Peter de Sèvres: consultant mnemonicist.'"

But if the object is a leaflet, say, EXAMINE should only describe the cover: READ would be the command a player would expect to use to get at the text itself. Inform normally defines READ to be the same command as EXAMINE, which is good for things like the business card, but counter-productive here. **The Trouble with Printing** shows how to separate these two commands, allowing any thing to have a property called its "printing" for text produced by READ, which will be different from its "description", the text produced by EXAMINE.

If the object is a lengthy diary, say, nobody would read it from cover to cover in a single IF turn. We might then want to allow the player to turn the pages one by one, with commands like READ PAGE 4 IN DIARY or READ THE NEXT PAGE: see **Pages**.

If the object is an encyclopaedic reference work, the player would consult named entries: see **Costa Rican Ornithology**, which allows commands like LOOK UP QUETZAL IN GUIDE.

Still larger sources of text often occur in IF: libraries or bookshelves, where many books are found together, and it is clumsy to write them as many individual items. One approach is to simulate an entire bookshelf with a single thing: see **Bibliophilia**. (This is much like looking up topics in a single book, except that each topic is a book in itself.) Another is to provide each book as an individual item, but have them automatically join together into a single portable collection: see **AARP-Gnosis**.

Signs, leaflets and encyclopaedias, being printed, have a wording which will never change during play. But sometimes the player reads something which acts of its own accord. Text substitutions are usually all that is needed to achieve this:

> The computer display is on the desk. The description is "Giant green digits read: [the time of day]."

This is easy because we know all the variations we want. But what if we want the player to write his own text, for instance, adding to a diary? This is trickier, because it means storing text as the player typed it, and replaying it later. (And suppose the player types reams and reams of text, not just a few words as we might have hoped?) **The Fourth Body** and **The Fifth Body** show how to use an external file - a multimedia trick requiring features only available if the project is set to the Glulx story file format - to handle even the most difficult cases.

Should we want a computer that responds to vocal commands, as in ASK COMPUTER ABOUT KLINGONS, the built-in extension Inanimate Listeners will allow the player to talk to inanimate objects as well as people.

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.5. Dice and Playing Cards

Onward to §9.7. Painting and Labeling Devices

Example 292: ★ **The Trouble with Printing** Making a READ command, distinct from EXAMINE, for legible objects.

Example 303: ★ **Pages** A book with pages that can be read by

number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

Example 443: ★ **The Fourth Body** Notebooks in which the player can record assorted notes throughout play.

Example 324: ★★ **Bibliophilia** A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

Example 335: ★★ **AARP-Gnosis** An Encyclopedia set which treats volumes in the same place as a single object, but can also be split up.

Example 444: ★★ **The Fifth Body** An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

Example 94: ★★★ **Costa Rican Ornithology** A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

§9.7. Painting and Labeling Devices

Writing on something is only one way a player can change its visual appearance. IF authors have long been wary of paint brushes, because a sufficiently motivated player could go through a whole landscape like a graffiti artist with a railway bridge. We want to give the player the illusion of freedom of action, while avoiding a situation where unlimited numbers of different decorations might be needed - that would need a table of potentially unlimited size.

One approach is to limit the number of items which can be decorated. In **Palette**, only the canvas can be painted, and each image overlays the last. **Early Childhood** increases the range to allow a whole kind ("block") to be painted, and also shows how the changing colours can be used to distinguish between otherwise identical objects.

Brown finds a different way to limit the number of simultaneous decorations: almost anything can have a red sticky label attached, but there is only one red sticky label. (So to decorate a new item, the player must first un-decorate an old one.)

See **Electricity and Magnetism** *for another form of stickiness*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.6. Reading Matter

Onward to §9.8. Simple Machines

Example 308: **Palette** An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean") are accepted alongside everyday ones ("blue").

Example 36: **Brown** A red sticky label which can be attached to anything in the game, or removed again.

Example 55: **Early Childhood** A child's set of building blocks, which come in

three different colours - red, green and blue - but which can be repainted during play.

§9.8. Simple Machines

The "device" kind provides for the simplest form of machine: one which is, at any given moment, switched on or switched off. Inform looks after this state, but leaves it to us to make the machine actually do something:

```
The air-conditioning unit is a device in the Florist's Shop. The air-
conditioning is fixed in place and switched on.


Every turn when the location is the Florist's Shop:
    if the air-conditioning is switched off, say "You worry about the cut
flowers in this jungle-hot air.";
    otherwise say "There is an low susurration from the air-
conditioning unit."
```

One primary dictionary definition for a machine is "an apparatus using or applying mechanical power and having several parts", and we often use the "part of" relationship to build machinery. **Control Center** provides a neat way to display the component parts of a machine to the player who examines it.

One component almost always part of an electrical machine is the (literal) switch, lever or button to control whether it is switched on or off. In **Model Shop** just such an on/off button is automatically made part of every device.

While an electrical device has only two states, a mechanical machine might have many, and for these the best approach is to define a kind of value naming the possibilities: see **Signs and Portents**, where the states are the possible destinations pointed towards.

Perhaps stretching the definition of "machine", **What Makes You Tick** demonstrates a fishing pole which the player can put together from several pieces.

 See **Bags, Bottles, Boxes and Safes** *for a safe that can be dialed to different combinations*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books,

Electronics
Back to §9.7. Painting and Labeling Devices
Onward to §9.9. Televisions and Radios

Example 61: ★ **Control Center** Objects which automatically include a description of their component parts whenever they are examined.

Example 427: ★ **What Makes You Tick** Building a fishing pole from several component parts that the player might put together in any order.

Example 57: ★★ **Model Shop** An "on/off button" which controls whatever device it is part of.

Example 53: ★★★ **Signs and Portents** Signpost that points to various destinations, depending on how the player has turned it.

---

§9.9. Televisions and Radios

IF authors often provide clues or background information to the player by means of radio broadcasts, TV shows or video tapes because they can talk to the player without needing to listen back, or to react to what the player does. The simplest radio set, like the one in **Aftershock**, really only has one thing to say: which is serendipitously being broadcast just at the moment the player tunes in (regardless of when that is). The next simplest approach is to spool a broadcast on an endless loop taking several turns to play through, as in **Radio Daze**.

Televisions come in all shapes and sizes, and **Aspect** allows their shape ("aspect ratio") to be described by the player.

In **Channel 1**, we can also refer to the television by what it is currently showing: thus WATCH CHANNEL 4 will work if the set is indeed tuned to 4. In **Channel 2**, numbered channel changing is taken further: we can now TUNE TV TO CHANNEL 3, as well. **Channel 2** is a reasonable base implementation of a television for many purposes.

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics
Back to §9.8. Simple Machines
Onward to §9.10. Telephones

Example 66: ★ **Radio Daze** A radio that produces a cycle of output using varying text.

Example 311: ★ **Aspect** Understanding aspect ratios (a unit) in the names of televisions.

Example 313: ★★ **Channel 1** Understanding channels (a number) in the names of televisions.

Example 316: ★★★ **Channel 2** Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

Example 336: ★★★ **Aftershock** Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times.

§9.10. Telephones

Telephones are much harder to achieve than televisions and in some ways as difficult to make convincing as a human character is - though of course there are corners which can be cut: we could have the reception drop off, or the other party hang up in a misunderstanding, and so on.

A single telephone line is tricky enough to provide that one might just as well have a general solution providing a whole network. **Four Cheeses** demonstrates a system where we can dial either people or numbers: CALL JANET ON TELEPHONE, or

CALL 7103, for instance.

While **Four Cheeses** provides only four-digit phone numbers, like internal company extensions, **Alias** shows how to manage US-style seven digit numbers.

Finally, we might occasionally want the player to be able to address a microphone or telephone receiver directly when the identity of the person on the other end is unknown, in the form TELL MICROPHONE ABOUT CRIME. Ordinarily Inform will disallow this because we're not allowed to talk to inanimate objects, but the extension Inanimate Listeners provides more options.

See **Saying Complicated Things** *for more approaches to conversation*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.9. Televisions and Radios

Onward to §9.11. Clocks and Scientific Instruments

Example 226: ★★★ **Four Cheeses** A system of telephones on which the player can call distant persons and have conversations.

Example 253: ★★★ **Alias** A telephone with phone numbers of the standard American seven-digit length.

§9.11. Clocks and Scientific Instruments

The simplest form of clock is a wrist watch. Here is a choice of analogue or digital:

```
The player wears a wrist watch. The description of the wrist watch is
"It is [the time of day in words]."


The player wears a digital watch. The description of the digital watch
```

is "It is [the time of day]."

Better clocks would allow us also to set the time, and to stop and start them: see **Tom's Midnight Garden**.

Scientific instruments provide sharper versions of our own senses. In the case of vision, they allow us to see closer up, or further away. It's a convention of IF that people can normally see only the current location, that is, they cannot see from one location into another. The boundary of the current room is like a horizon, even out of doors (though it's true that there are ways to disguise that with a continuous outdoor landscape). **Ginger Beer** provides a telescope able to see into other rooms.

Witnessed 2 provides a meter which measures how close a ghost is to the player.

See **Continuous Spaces and The Outdoors** *for more on seeing into adjacent locations*

See **Heat** *for infrared goggles*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

Back to §9.10. Telephones

Onward to §9.12. Cameras and Recording Devices

Example 180: **Witnessed 2** A piece of ghost-hunting equipment that responds depending on whether or not the meter is on and a ghost is visible or touchable from the current location.

Example 299: **Tom's Midnight Garden** A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running. Time on the face is also reported differently depending on whether the clock is analog or digital.

Example 365: ★★ **Ginger Beer** A portable magic telescope which allows the player to view items in another room of his choice.

---

§9.12. Cameras and Recording Devices

Recording what is going on, for later playing back or examination, is difficult because the range of situations is very complex. Exactly how much information should we store when we make a recording, and will this require problematically large tables? Will it be difficult even to do at all?

The usual approach is to record only basic details of events or situations. In **If It Hadn't Been For...** the tape recorder preserves only a few different sounds - footsteps, creaking, rustling - rather than capturing exactly the sound of every action taking place in earshot. In **Claims Adjustment**, we can take up to 36 Polaroid-style photographs, but each is described only by saying what it is a photo of. Thus we can have a photograph of a vase, or even a photograph of a photograph of a vase (because that too is a thing), but not a photograph of a still life in which several items have been gathered together by the player. That would ordinarily require too much storage.

A similar trick, though involving impromptu sculpture rather than photography, can be found in **Originals**. (The artist magically "manifests" these models rather than sculpting the conventional way in order to avoid the nuisance of carrying around raw materials - wax maquettes and so forth - which would clutter up the example.)

Text, of course, can store arbitrary descriptions. **Mirror, Mirror** provides a perfect visual recorder: it remembers a room description exactly as the player saw it at the time.

Actor's Studio provides a video camera that records and time stamps all actions performed in its presence while it is set to record.

See **Telephones** *for ways to speak to inanimate objects, which might be appropriate when, say, tape-recording a confession*

Start of Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics
Back to §9.11. Clocks and Scientific Instruments
Onward to Chapter 10: Physics:

Substances, Ropes, Energy and Weight: §10.1. Gases

Example 273: ★ **If It Hadn't Been For...** A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

Example 369: ★ **Originals** Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

Example 417: ★ **Mirror, Mirror** The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

Example 221: ★★ **Actor's Studio** A video camera that records actions performed in its presence, and plays them back with time-stamps.

Example 322: ★★ **Claims Adjustment** An instant camera that spits out photographs of anything the player chooses to take a picture of.

Examples from Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

48 **Example** **Would you...?** *WI*
Adding new properties to objects, and checking for their presence.

For instance, if we want to give some objects a flavor:

```
"Would you...?"

The House is a room. The mouse is an animal in the
House.

The player carries some green eggs and a ham.

A food is a kind of thing that is edible. Food has some
text called flavor. The flavor of food is usually
"Tolerable."
```

Things are, in general, not edible by default, so we have to make them edible specifically in order to allow them to be eaten by the player. Here we've defined food to be edible by default, and we have given it a standard piece of flavor text.

```
The ham and the green eggs are food. The flavor of the
green eggs is "Delicious!"

After eating something:
    if the noun provides the property flavor, say "[the
flavor of the noun][paragraph break]";
```

```
otherwise say "It's [noun]-flavored."
```

Note that we use "if the noun provides a flavor..." to make sure that the property exists before attempting to use it. Otherwise, there is the risk that we will try to print a property that does not exist, resulting in errors in the game.

We will only get the "It's [noun]-flavored." response if we successfully eat something that is not a food and does not have flavor text. To test this feature, let's suppose something that isn't exactly food but can theoretically be chewed on:

```
The player carries some paper. The paper is edible.
```

```
Test me with "eat ham / eat green eggs / eat paper".
```

---

133 **ExampleCandy** *WI*

One of several identical candies chosen at the start of play to be poisonous.

Suppose we want to give the player a bag of candies, of which a random one is poisonous. We can pick which one should be poisoned at the start of play, like this:

```
"Candy"
```

```
The plural of piece of candy is pieces of candy. A piece of candy is a kind of thing. A piece of candy is always edible. Four pieces of candy are in the Halloween bag.
```

```
Toxicity is a kind of value. The toxicities are safe and poisonous. A piece of candy has a toxicity. A piece of candy is usually safe.
```

```
The Porch is a room. The player carries the Halloween
bag.


After eating a poisonous piece of candy:
    say "Oh, that didn't taste right at all. Oh well!"


When play begins:
    now a random piece of candy is poisonous.


Test me with "eat candy / g / g / g".
```

Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

Many older interactive fiction games required the player to find food and eat on a regular basis in order to avoid death. This effect was often unrealistic (since most people can survive much longer than a few hours without eating) and is often seen as an annoyance. However, for the sake of argument, suppose that we do want to construct a hunger-and-death system.

To make things a little more interesting, we will postulate that different foods are differently filling, so that if the player manages to find something really caloric, he is off the hook on his hunger search for a while.

We will also implement the system so that the player gets messages when he is hungry, then dies a short time later. (The times involved are ludicrously short, but this allows us to see the effects within a simple example. In a real game we would want to allow a considerably longer timer for the hunger to play out.)

First, a little scene-setting:

```
"MRE"

When play begins:
    now the right hand status line is "[time of day]";
    say "The procedure was painless at first: increased
strength was the first sign, followed by a sensation of
delayed time, as though everyone around you moved
more slowly. Your ability to dodge and perform feats of
agility doubled, then trebled. You were heralded as a
triumph of medicine. They told you there would be no
side effects worth speaking of.

They were wrong."

The Base Camp Larder is a room. "This room has been
reinforced after each incident -- and there have been
dozens in the last two months -- so that it now rivals Fort
Knox. Only your new skill and speed enabled you to
dodge the motion sensors, knock out the computerized
security system, fool the retinal scanner, and punch a
hole in the steel containment grating. But you're inside
now."
```

Now we define our food, and add some special instructions for what happens to our hunger counters when the food is eaten:

```
Food is a kind of thing. Food is usually edible. Food has
a time called the satisfaction period. The satisfaction
period of a food is usually 5 minutes.

A person can be hungry or replete. The player is
hungry.

The Larder contains an apple, a candy bar, and a large
plate of pasta. The apple, the candy bar, and the pasta
are food. The satisfaction period of the apple is 2
```

```
minutes. The satisfaction period of the pasta is 125
minutes.


Check eating something which is not food:
    say "[The noun] might be edible, but it's not what
you'd consider really food."


Check eating something when the player is not hungry:
    say "You're not hungry right now."


Carry out eating something:
    now the player is replete;
    hunger resumes in the satisfaction period of the noun
from now.
```

The first of those two phrases, "now the player is replete", causes the player to cease to be hungry; the second one sets up a future event in which the hunger sets in again. The length of time until that event depends on how satisfying the specific food is. Now we define that event:

```
At the time when hunger resumes:
    starvation occurs in three minutes from now;
    now the player is hungry.


At the time when starvation occurs:
    if the player is hungry, end the story saying "You
have starved".
```

Note "if the player is hungry": it is possible that the starvation event will be set up but the player will eat before it occurs; in that case, we want it not to take effect.

And now, since we really ought to give the player some warning of what is happening to him:

```
Every turn when the player is hungry:
```

```
    choose a random row in the Table of Hunger
Complaints;
    say "[hunger entry][paragraph break]".
```

```
Table of Hunger Complaints
hunger
"Gosh, you're starving."
"It feels as though you haven't eaten in days. Weeks,
almost."
"The world seems to slow down and everything
becomes sharper and brighter. You are a hunter, a
hunter of foodstuffs."
"You find yourself staring at [the random visible thing
that is not the player] and wondering how it would
taste."
```

```
Test me with "eat apple / z / z / z / eat candy bar / z / z /
z / z / z / z / z / z / z".
```

---

391 ExampleStone *WI*

A soup to which the player can add ingredients, which will have different effects when the player eats.

A thing can have a rule as a property, if we like. Here we are going to allow the player to make a soup whose effects will depend on its ingredients. Each ingredient will have its own "food effect" rule, to be followed when the food is eaten.

Note that there are other, slightly less cumbersome ways to do the same thing -- we will see in the chapter on Rulebooks that we could make a "food effects rulebook" and then write a number of rules such as "food effects rule for carrots" or "food effects rule for the stone". Nonetheless, we demonstrate rules-as-properties here for the sake of thoroughness.

So:

```
"Stone"
```

```
A food is a kind of thing that is edible. A food has a rule
called the food effect.



Carry out eating a food:
    if a food is part of the noun:
        repeat with item running through things which are
part of the noun:
            if item is a food, follow the food effect of the
item;
    follow the food effect of the noun.



Report eating a food:
    say "You eat [the noun]. [diagnosis of the player]";
    stop the action.



To say diagnosis of (victim - a person):
    if the victim is ill:
        say "You are ill.";
        rule succeeds;
    otherwise:
        say "You are healthy. ";
    if the victim is awake, say "You are wide awake. ";
    otherwise say "You are sleepy. ";
    if the victim is bright-eyed, say "Your eyesight is
clear. ";
    otherwise say "Your eyesight is dim. ";
    if the victim is weak, say "You are weak. ";
    otherwise say "You are strong. ";
    if the victim is hungry, say "You are hungry.";
    otherwise say "You are well-fed."
```

And now to provide some particular foods:

```
Some carrots are a food. The food effect of carrots is
the bright-eye rule. This is the bright-eye rule: now the
player is bright-eyed.
```

Some potatoes are a food. The food effect of the potatoes is the sleepiness rule. This is the sleepiness rule: now the player is sleepy.

The broth is a food. The indefinite article of the broth is "some". The food effect of broth is the filling rule. This is the filling rule: now the player is full.

The hambone is a food. The food effect of the hambone is the heartiness rule. This is the heartiness rule: now the player is strong. Instead of eating the hambone: say "You cannot just eat a bone!"

The poison ivy is a food. "Poison ivy grows nearby." The food effect of poison is the illness rule. This is the illness rule: now the player is ill.

A person can be bright-eyed or blind. The player is blind.

A person can be well or ill. The player is well.

A person can be hungry or full. The player is full.

A person can be strong or weak. The player is weak.

A person can be awake or sleepy. The player is sleepy.

The broth is in the kettle. The kettle is on the fire. The

```
fire is in the Clearing. The Clearing is a room.


The player carries the hambone, the potatoes, and the
carrots. The ivy is in the clearing.


Instead of examining the broth:
    if something is part of the broth, say "In the broth, [a
list of things that are part of the broth] float[if exactly one
thing is part of the broth]s[end if].";
    otherwise say "It is just a thin broth with no other
ingredients."


Instead of inserting something into the broth: try
inserting the noun into the holder of the broth.


Instead of taking the broth: say "You cannot take the
broth in your bare hands."
```

And the following is a relatively unimportant nicety:

```
To sink is a verb.


After inserting a food which is not the broth into a
container which contains the broth:
    now the noun is part of the broth;
    say "[The noun] [sink] into [the second noun], making
the broth richer."


Test me with "x broth / eat hambone / put hambone in
kettle / x broth / put potatoes in broth / x broth / eat
carrots / eat broth / put ivy in kettle / eat ivy".
```

 ExampleDelicious, Delicious Rocks *WI*

Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

In some cases, we may want to add new stages to action processing. One possibility is a stage where we check the sanity of what the player is trying to do before executing any of the other commands; so that we avoid, for instance

```
>EAT ROCK
(first taking the rock)
That's plainly inedible.
```

Here is how we might insert such a stage in our action processing, using rulebook manipulation.

```
"Delicious, Delicious Rocks"


Section 1 - Procedure


The sanity-check rules are a rulebook.


This is the sanity-check stage rule:
    abide by the sanity-check rules.


The sanity-check stage rule is listed after the before
stage rule in the action-processing rules.


Section 2 - Scenario
```

```
Candyland is a room. The lollipop tree is an edible thing
in Candyland. The genuine rock is a thing in Candyland.


Sanity-check eating an inedible thing:
    say "Your digestion is so delicate -- you're sure [the
noun] wouldn't agree with you." instead.


Test me with "eat lollipop / eat rock".
```

Notice that now Inform does not try taking the rock before rejecting the player's attempt to eat it.

It is of course possible to get the same effect with

```
Before eating an inedible thing:
    say "Your digestion is so delicate -- you're sure [the
noun] wouldn't agree with you." instead.
```

...and in a small game with few rules, there's not much reason to add an extra stage. The ability to modify the stages of action processing becomes useful when we have a fairly large game with sophisticated modeling and want to be sure that some kinds of message (such as the sanity-check) are always handled before other things that we might be doing at the before stage (such as generating implicit actions like opening doors before going through them).

ExampleSafety

A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.


```
"Safety"


The Vault is a room. "Snug yet paranoid, this represents
the state of the art in cheerless security." The Safe is
```

here. "A mammoth safe, with a dial which can spin to any number, has pride of place. It must weigh about the same as a small car, so don't get any ideas." Instead of opening the safe, say "The safe opens only when turned to the correct combination."

In the Safe is a silver florin. The Safe is closed and fixed in place. Understand "dial" as the Safe.

Spinning it to is an action applying to one thing and one number. Check spinning it to: if the noun is not the Safe, say "[The noun] does not spin." instead. Report spinning it to: say "Click! and nothing else happens."

Understand "spin [something] to [a number]" as spinning it to.

After spinning the closed Safe to 1384: now the Safe is open; say "Clonk! and the safe door swings slowly open, revealing [a list of things in the Safe]."

Test me with "open safe / spin safe to 1131 / open safe / spin safe to 1384 / x safe / get florin".

ExampleCinco

A taco shell that can be referred to (when it contains things) in terms of its contents.

It's fairly common that we want to be able to refer to a container in terms of what it has in it: a bottle of wine, a salt shaker, a chicken sandwich. The player is free to remove the contents again, and the object will go back to using its usual name:

"Cinco"

Cinco de Mayo Fundraiser is a room.

The taco shell is an edible thing in the Fundraiser. It is a portable container. It has carrying capacity 1.

Understand "[something related by containment] taco" as the taco.

Rule for printing the name of the taco shell while not inserting or removing:
    if the taco contains something (called filling), say "[filling] taco";
    otherwise say "taco shell";
    omit contents in listing.

The player carries shredded beef. It is edible.

The taking action has an object called source (matched as "from").

Setting action variables for taking:
    now source is the holder of the noun.

Report taking something from the taco shell:
    say "You gingerly pick [the noun] out of the taco shell." instead.

Test me with "x taco / put shredded beef in taco / get taco / i / x shredded beef taco / get shredded beef / x shredded beef taco".

---

339 Example Shipping Trunk *WI*

A box of baking soda whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

First to lay some groundwork:

"Shipping Trunk"

A chest is a kind of container. A chest is always openable. A lid is a kind of supporter. A lid is part of every chest.

Before opening a chest when something (called the obstruction) is on a lid which is part of the noun:
    say "Better remove [the obstruction]." instead.

A thing can be innocent or smelly.

The Storage Unit is a room. The shipping trunk is a closed chest in the Storage Unit. The trunk contains some garlic, a loaf of moldy sourdough, a mildewy bathtowel, a pair of unwashed socks, two dead trout, and a box of baking powder. The garlic, trout, sourdough, bathtowel, and socks are smelly. The baking powder is innocent.

The shipping trunk's lid supports a small card. The

```
description of the small card is "'Please, please do not
open this trunk.'"



After opening the trunk:
    if the trunk had been open:
        say "You steel yourself...";
        continue the action;
    otherwise:
        say "There roils up from inside an indescribable
funk, which, when you can see straight, you have no
trouble attributing to the presence of [a list of smelly
things in the trunk]. You also note [a list of innocent
things in the trunk] in the corner.".
```

And now, with that preparation:

```
Before printing the name of the baking powder when the
powder is in a container which contains a smelly thing:
say "completely ineffective ".



Test me with "open trunk / examine card / get card /
open trunk / get powder / inventory".
```

---

344 **ExampleUnpeeled** *WI*

Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

```
"Unpeeled"



Scullery is a room. A sack is carried by the player. The
sack contains a yellow onion. The player carries a cork.
```

```
Before printing the name of the onion while listing
contents:
    if the holder of the onion contains exactly 1 thing, say
"single ".


Test me with "i / put cork in sack / i".
```

---

433 **Example** **Eyes, Fingers, Toes** *WI*

A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

It is not difficult to implement a safe which can be set to a single number to open; but a more common scenario in the real world is for the safe to open on a sequence of numbers when they have been dialed in the right order.

For IF, this means that we have to keep running track of the last N digits the player has dialed, dropping the first digit and adding a new one to the end each time the player re-dials the safe. This is a perfect occasion for lists:

```
"Eyes, Fingers, Toes"


The Addams Wine Cellar is a room. It contains a closed
lockable locked container called a safe.


The safe has a list of numbers called the current
combination.


The safe has a list of numbers called the true
combination. The true combination of the safe is {2, 10,
11}.
```

```
Understand "set [something] to [a number]" as setting it
numerically to. Setting it numerically to is an action
applying to one thing and one number.



Instead of examining the safe:
    if the number of entries in the current combination of
the safe is 0,
        say "You haven't dialed the safe to any
combination yet.";
    otherwise say "You have dialed the safe to [the
current combination of the safe].".



Check setting something numerically to (this is the block
setting numerically rule):
    say "[The noun] cannot be set."



Instead of setting the safe numerically to the number
understood:
    truncate the current combination of the safe to the
last 2 entries;
    add the number understood to the current
combination of the safe;
    if the safe is locked and the current combination of
the safe is the true combination of the safe:
        say "You dial [the number understood], and [the
safe] gives a joyous CLICK.";
        now the safe is unlocked;
    otherwise if safe is unlocked and the safe is closed
and the current combination of the safe is not the true
combination of the safe:
        say "You spin the dial, and [the safe] snicks
locked.";
        now the safe is locked;
    otherwise:
        say "You dial [the number understood] on the
safe."



Test me with "x safe / set safe to 10 / x safe / set safe to
29 / x safe / set safe to 2 / x safe / set safe to 10 / x safe
```

/ set safe to 11 / open safe / set safe to 14 / close safe / set safe to 15 / open safe".

 Example **Trachypachidae Maturin 1803** *WI*

Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

"Trachypachidae Maturin 1803"

A bottle is a kind of container. Bottles are usually openable, transparent, and closed. A cork is a kind of thing. A cork is in every bottle.

Understand "cork [something]" as corking.

Understand the command "stopper" as "cork".

Understand "uncork [something]" as uncorking.

Corking is an action applying to one thing.

Check corking:
    if the noun is not a bottle, say "[The noun] cannot be corked." instead.

Carry out corking:
    try closing the noun.

Uncorking is an action applying to one thing.

Check uncorking:
    if the noun is not a bottle, say "[The noun] cannot be uncorked." instead.

Carry out uncorking:
    try opening the noun.

Understand "close [something] with [something preferably held]" as corking it with.

Understand "cork [something] with [something preferably held]" as corking it with.

Corking it with is an action applying to one thing and one carried thing.

Check corking it with:
    if the noun is not a bottle, say "[The noun] cannot be corked." instead;
    if the second noun is not a cork, say "[The second noun] will not fit in [the noun]." instead.

Carry out corking it with:
    try inserting the second noun into the noun instead.

Instead of closing a bottle:
    if a cork (called the item) is carried by the player, try inserting the item into the noun instead;
    otherwise say "You need a stopper of some kind."

Instead of opening a bottle:
    if a cork (called the item) is in the noun, try taking the item instead;
    otherwise say "[The noun] has no stopper."

Carry out inserting a cork into a bottle:
    now the second noun is closed.

After inserting a cork into a bottle:
    say "You stopper [the second noun] with [the noun]."

Before taking a cork when the noun is in a closed bottle (called the item):
    now the item is open.

Instead of taking a cork when the noun is in a bottle (called the item):
    move the noun to the player;
    say "You pull [the noun] from [the item]." instead.

Before printing the name of a bottle (called target) while not inserting, taking, searching, or removing:
    if the target is closed, say "sealed ";
    otherwise say "now open ".

After printing the name of a bottle (called target) while not inserting, searching, examining, or removing:
    if the target contains a noncork thing, say " containing [a list of noncork things in the target]";

omit contents in listing.

Instead of examining a bottle:
say "[The noun] contains [a list of noncork things in the noun]."

Definition: a thing is noncork if it is not a cork.

The Doctor's Cabin is a room. "A dark, cramped triangle, like a slice of cake, except that its sharp end has been cut off: and so low that a moderately tall man would strike his head on the deck above if he were to stand upright. Every free surface is covered with sheets of best Venetian looking-glass, to increase the light filtering in. Long use and the carpenter's ingenuity have packed in a folding cot and table, and lockers are built into unlikely places: lockers filled with specimens, skeletons, sketches, drafts and serial letters." The jug is a bottle in the Doctor's Cabin. The jug contains a beetle. The description of the beetle is "The doctor assures you that it is a nondescript."

Test me with "get jug / x jug / open jug / x jug / i / x cork / cork jug / i / uncork jug / i / x jug / get beetle / i / close jug / i / x jug".

---

342 **ExampleHudsucker Industries** *WI*

Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

In this scenario, the player starts with a bag full of unsorted letters. These can be polite or rude, but he won't know which until he has examined them. What's more, he is allowed to sort the letters, in which case a group of letters will be shown as (for instance) "two polite letters"; but a group of mixed letters, even if they have all been read, will be called "unsorted letters".

Further, the player should be allowed to refer to sorted letters by tone, but not unsorted letters.

To do this, we'll need printing the name... and printing the plural name..., as well as some special understanding rules.

```
"Hudsucker Industries"


Tone is a kind of value. The tones are effusive,
affectionate, polite, curt, and flamingly rude.


A letter is a kind of thing. The description of a letter is
usually "On inspection, it turns out to be quite [tone]." A
letter has a tone. The tone of a letter is usually polite.


A letter can be read or unread. A letter is usually
unread. Carry out examining a letter: now the noun is
read.


Before printing the name of a read letter: say "[tone] ".


Before printing the name of an ungrouped letter: say
"random ".


Before printing the plural name of a letter (called the
subject):
    if the subject is grouped:
        say "[tone] ";
    otherwise if the number of unread letters which are
next to the subject is 0:
        say "unsorted ".
```

After printing the plural name of a letter (called the subject):
    if the number of read letters which are next to the subject is 0, say " (all unread, at the moment)" instead;
    if the number of unread letters which are next to the subject is greater than 0, say " (some as yet unread)" instead.

Proximity relates a thing (called X) to a thing (called Y) when the holder of X is the holder of Y. The verb to be next to means the proximity relation.

Definition: a letter is grouped:
    if it is unread, no;
    if the number of unread letters next to it is greater than 0, no;
    repeat with item running through letters which are next to it:
        if the tone of item is not the tone of it, no;
    yes.

Definition: a letter is ungrouped if it is not grouped.

The Mailroom is a room. "Usually a thrumming hive of bee-like workers, but you got in early to get a jump on the day's work."

The satchel is carried by the player. Two flamingly rude letters are in the satchel. Five polite letters are in the satchel.

The mail wall are fixed in place in the mailroom. "Before you is a wall of mailboxes, including [a list of mailboxes which are part of the mail wall]."

```
The plural of mailbox is mailboxes. A mailbox is a kind
of container. The CEO box is a mailbox. The Hold box is
a mailbox. The Trash box is a mailbox. Understand
"mailbox" as a mailbox.
```

Now, there's a good bit of interaction to streamline. We intend that the player will be taking letters from the satchel, reading them, and putting them (perhaps grouped) into boxes. Our interaction rules should assist in this process as much as possible. To start with, the player will be most likely to examine letters he hasn't read yet:

```
Does the player mean examining a letter (called the
subject):
    if we have examined the subject, it is very unlikely;
    it is very likely.
```

The rules about taking are more subtle: the player is more likely to want to take an ungrouped letter than a grouped one; he is more likely to want one from the satchel than not; and he is most unlikely to want to take a letter (grouped or ungrouped) that he is already holding.

```
Does the player mean taking a letter (called subject)
which is grouped:
    if the player carries the subject, it is very unlikely;
    if the subject is in the satchel, it is possible;
    it is unlikely.
```

```
Does the player mean taking a letter (called subject)
which is ungrouped:
    if the player carries the subject, it is very unlikely;
    if the subject is in the satchel, it is very likely;
    it is possible.
```

And finally, we will assume by default that anything other than examining or taking is most likely to apply to a letter he's already identified:

```
Does the player mean doing something other than
examining or taking with a letter (called the subject):
    if we have examined the subject, it is likely;
    it is unlikely.
```

And we would also like to understand properties under the same circumstances as printing -- a letter will be identifiable as "polite" if it's already been read and it is either by itself or in a sorted stack of polite letters, but otherwise not. What's more, to make it possible to disambiguate commands in the other direction, we'll call any unsorted letter "random", to represent that the player doesn't know what it is.

```
Understand the tone property as referring to a letter
when the item described is grouped. Understand
"random" as a letter when the item described is
ungrouped.


When play begins:
    now every mailbox is part of the mail wall;
    repeat with switch count running from 1 to 5:
        move a random letter to the satchel.


Test me with "inventory / examine letter / get letter / i /
put letter in ceo box / inventory / get letter / x letter / g / g
/ i / x letter / g / g / i / put letter in hold box / get letter / g /
g / i".
```

That last "repeat" is merely a device to shuffle the order of items in the satchel so that the player will not always encounter the letters in a neatly presorted order, despite our defining them that way. (Of course, that means that the test produced by TEST ME cannot be very exciting...)

---

35 **Example** **Fallout Enclosure** *WI*

Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

It may not be immediately obvious why we might want to create new intermediate categories of the kinds hierarchy. But there may be times, for instance, where we would like to make an action that applies in the same way to both containers and supporters, but to nothing else in the game. To avoid creating two nearly-identical rules, we would instead roll the two categories together into one, on the principle that duplicating source text is usually a sign of bad design.

So for instance let's say the player is able to zap objects to make them go away, but any contents -- things inside a container or on top of a supporter -- should always be left as residue. Here's one way we might do this:

```
"Fallout Enclosure"


Section 1 - Procedure


An enclosure is a kind of thing. A container is a kind of
enclosure. A supporter is a kind of enclosure.


Understand "zap [something]" as zapping. Zapping is an
action applying to one thing. The Zapping action has a
list of things called the remnants.


Carry out zapping an enclosure:
    if the noun holds something:
        now the remnants is the list of things held by the
noun;
        repeat with N running through the remnants:
            move N to the holder of the noun.


Carry out zapping:
    now the noun is nowhere.


Report zapping:
    say "You zap [the noun], destroying [them][if the
remnants is not empty] and leaving [the remnants with
indefinite articles] behind[end if]."
```

Section 2 - Scenario

SuperDuperMart is a room. SuperDuperMart contains some shelves and a cash register.

The shelves support a bottle of Buffout and a container of Jet.

The cash register contains some prewar money, a coin purse, and a bottle cap. The coin purse contains a prewar nickel. It is closed.

The cash register is closed and locked.

Test me with "zap shelves / zap buffout / zap register / zap purse".

ExampleBeing Prepared

A kind for jackets, which always includes a container called a pocket.

"Being Prepared"

A jacket is a kind of thing. A jacket is always wearable.

A pocket is a kind of container. A pocket is part of every jacket. The carrying capacity of a pocket is always 2.

```
After examining a jacket:
    let target be a random pocket which is part of the
noun;
    say "[The target] contains [a list of things in the
target]."
```

Now we've created the rules that will govern any specific jackets we might happen to put in our game: each one will always have one pocket, which will be able to contain no more than two things. The description of "a list of things" is text with a list, which we will learn about further in a few sections.

Next we might want to create the environment and an actual example of the jacket kind:

```
Tent is a room. "A dome made of two flexible rods and a
lot of bright green ripstop nylon. It bills itself as a one-
man tent, but you'd call it a two-dwarf tent: there is no
way to arrange yourself on its square floor so that you
can stretch out completely."


The hoodie is a jacket. "Your hoodie is balled up in the
corner." The description of the hoodie is "Both elbows
are stained from yesterday's entrenching project."


The hoodie's pocket contains a Swiss army knife and a
folded map. The hoodie is in the Tent.
```

Notice that, since Inform has created a pocket for the hoodie, we can now refer to it by name in our source, giving it any additional properties we need to define. Here we simply put a few items into it.

```
The player wears a whistle. The description of the
whistle is "To frighten bears."


Test me with "x hoodie / get hoodie / get knife / get map
```

```
/ i / put hoodie in pocket / put whistle in pocket / put map
in pocket / put knife in pocket / i".
```

Notice that Inform automatically refuses to put the hoodie into its own pocket: as a default, a container cannot contain something of which it is itself a part.

---

332 Example Some Assembly Required *WI*

Building different styles of shirt from component sleeves and collars.

We now have the mechanisms in place to do some fairly sophisticated renaming of objects. For instance:

```
"Some Assembly Required"

Garment type is a kind of value. The garment types are
vest, t-shirt, polo shirt, mandarin blouse, button-down,
shell, experiment.

Every turn:
    assign identities.

When play begins: assign identities.

To assign identities:
    repeat with item running through torsos:
        reassess item.

To reassess (item - a torso):
    if the number of things which are part of the item is 0:
        now garment type of the item is vest;
```

```
        rule succeeds;
    if exactly two short sleeves are part of the item:
        if a collar is part of the item,
            now garment type of the item is polo shirt;
        otherwise now garment type of the item is t-shirt;
        rule succeeds;
    if exactly two long sleeves are part of the item:
        if a collar is part of the item,
            now garment type of the item is button-down;
        otherwise now garment type of the item is
mandarin blouse;
        rule succeeds;
    if a collar is part of the item and the number of
sleeves which are part of the item is 0, now garment
type of the item is shell;
    otherwise now garment type of the item is
experiment.



Before cutting something which is worn by the player:
    try taking off the noun.



Instead of cutting something when something is part of
the noun:
    say "You cut up [the noun], snipping off [a list of
things which are part of the noun].";
    now every thing which is part of the noun is in the
holder of the noun.



Instead of cutting something which is part of something:
    say "You carefully snip [the noun] free.";
    now the player carries the noun.



Rule for printing the name of a torso: say "[garment
type]".



A torso is a kind of thing. A torso is always wearable.
Understand "shirt" or "blouse" as a torso. A torso has a
garment type. Understand the garment type property as
describing a torso. A sleeve is a kind of thing. A short
```

```
sleeve is a kind of sleeve. A long sleeve is a kind of
sleeve. A collar is a kind of thing.


Understand "sew [something] to [something]" as affixing
it to. Affixing it to is an action applying to two things.
Carry out affixing something to something: now the
noun is part of the second noun. Report affixing
something to something: assign identities; say "You sew
[the noun] on, creating [a second noun]." Understand
the command "stitch" as "sew".


Instead of affixing something to something when the
second noun is worn: say "You're wearing [the second
noun]!"


Instead of affixing a torso to something:
    if the second noun is a torso, say "Couture for
Siamese twins is a daring field, but a bit of a niche
market.";
    otherwise try affixing the second noun to the noun.


Instead of affixing a sleeve to something when at least
two sleeves are part of the second noun: say "[The
second noun] already sports [a list of sleeves that are
part of the second noun]."


Instead of affixing a collar to something when a collar is
part of the second noun: say "[The second noun]
already sports [a list of collars that are part of the
second noun]."


Instead of examining something when something is part
of the noun: say "Stitched to [the noun] [is-are a list of
things which are part of the noun]."
```

Here is where the issue of precedence arises. We want to encourage

Inform to select a cuttable object that is part of something else, rather than one of the spares:

```
Definition: a thing is removable if it is part of something.
Understand "cut [removable thing]" as cutting.


The Boutique is a room. "Still festively strewn with the
confetti and streamers of the Grand Opening party, and
still almost totally customer-free."


The player carries a torso. The player carries three short
sleeves. The player carries two long sleeves. The player
carries two collars.


Test me with "sew collar to shirt / i / sew short sleeve to
shirt / g / i / x polo shirt / cut collar / i / cut shirt / sew
long sleeve to shirt / i / sew long sleeve to shirt / i / sew
collar to shirt / g / i / wear button-down".
```

Clark Gable in a pin-striped suit and a pink thong.

The following burlesque was considered too much for the tender readers of Chapter 3, since it involved explicit use of listing and persuasion:

```
"Hays Code"


The Movie Set is a room. Clark Gable is a man in the
Movie Set. "Clark leans on a polystyrene pillar, wearing
[a list of unconcealed things worn by Clark] with his
usual aplomb." Persuasion rule for asking Clark to try
```

doing something: persuasion succeeds.

Clark is wearing a pin-striped suit and a pink thong.
Rule for deciding the concealed possessions of Clark: if the particular possession is the thong and Clark is wearing the suit, yes; otherwise no.

Test me with "clark, remove suit / look / clark, remove thong / look".

---

242 **Example** **What Not To Wear** *WI*

A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

"What Not To Wear"

Section 1 - Overlying and Underlying

We start by borrowing some of the same ideas from the Bogart example, but we're also going to make a kind called "garment-element". This kind will include both garments (objects of clothing) and body parts (things that can be covered by clothing); using it allows us to restrict the way our underlying and overlying relations apply, which will make them a bit faster at run-time.

A garment-element is a kind of thing.

Underlying relates various garment-elements to various garment-elements with fast route-finding. The verb to underlie means the underlying relation. The verb to be under implies the underlying relation.

```
Check taking off:
    if the noun underlies something (called the
impediment) which is worn by the player, say "[The
impediment] [are] in the way." instead.

Carry out taking off:
    now the noun is not underlaid by anything.

Report taking off something:
    say "[We] [are] now wearing [a list of uppermost
things worn by the player]." instead.

Definition: a garment-element is uppermost if it is not
under something opaque.
```

Here we've expanded on the previous ideas of 'uppermost' because it is possible for an upper layer to reveal what lies beneath: a tie, a clear plastic trenchcoat, an open-knit sweater, etc. We'll make such items transparent.

```
Before taking off something which underlies something
which is worn by the player:
    while the noun underlies something (called the
impediment) which is worn by the player:
        say "(first removing [the impediment])[command
clarification break]";
        silently try taking off the impediment;
        if the noun underlies the impediment, stop the
action.

Overlying relates various garment-elements to various
garment-elements. The verb to overlie means the
overlying relation.
```

Covering relates a garment-element (called A) to a garment-element (called B) when the number of steps via the overlying relation from A to B is greater than 0. The verb to cover means the covering relation.

Before wearing something when a garment which covers the noun is worn by the player:
    while the player wears a garment (called the impediment) which covers the noun:
        say "(first removing [the impediment])[command clarification break]";
        silently try taking off the impediment;
        if the player is wearing the impediment, stop the action.

Carry out wearing:
    repeat with hidden item running through things worn by the player:
        if the noun covers the hidden item, now the hidden item underlies the noun.

Instead of looking under something which is worn by the player:
    if something (called the underwear) underlies the noun, say "[We] [peek] at [the underwear]. Yup, still there.";
    otherwise say "Just [us] in there."

Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying [a list of things carried by the player][else][We]['re] empty-handed[end if][if the player wears something]. [We] [are] wearing [a list of uppermost garments worn by the player][end if]."

To peek is a verb.

Section 2 - Regional Coverage

Here we draw in the idea that different clothes go over different areas of the body, and that they should be in competition with each other only if both sets of clothes belong at the same level over the same body area.

```
Before wearing something:
    let N be the layering depth of the noun;
    repeat with item running through things worn by the
player:
        if the layering depth of the item is N and the item
covers a body-part which is covered by the noun:
            say "(first taking off [the item])[command
clarification break]";
            silently try taking off the item;
            if the player wears the item, stop the action.
```

This may seem like overkill, but it allows us to create garments that cover different subsets of the body -- pants and shirt vs. a dress, for instance.

```
To decide what number is the layering depth of (chosen
garment - a thing):
    let N be 0;
    if the chosen garment covers a body-part (called
base):
        let N be the number of steps via the overlying
relation from the chosen garment to the base;
    decide on N.
```

To help with modeling, we'll give everyone body parts, broken down according to their relevance to clothing:

```
A body-part is a kind of garment-element. A torso, a
seat, a head, pair of legs, and pair of feet are kinds of
body-part.
```

If we wanted to allow gloves, we might put in hands as well; but this is enough for now.

```
One head is part of every person. One torso is part of
every person. One pair of legs is part of every person.
```

One pair of feet is part of every person. One seat is part of every person.

And now we make some categories of clothing:

A garment is a kind of garment-element. A garment can be transparent. A pair of pants, a pair of underpants, a foundation garment, a pair of socks, a pair of shoes, a jacket, a hat, a dress, and a shirt are kinds of garment.

The plural of pair of pants is pairs of pants. The plural of pair of underpants is pairs of underpants. The plural of pair of socks is pairs of socks. The plural of pair of shoes is pairs of shoes.

A pair of pants, a pair of underpants, a foundation garment, a pair of socks, a pair of shoes, a jacket, a hat, a dress, and a shirt are usually wearable.

When play begins:
now every pair of socks overlies every pair of feet;
now every pair of shoes overlies every pair of socks;
now every pair of underpants overlies every seat;
now every pair of pants overlies every pair of underpants;
now every foundation garment overlies every torso;
now every jacket overlies every shirt;
now every jacket overlies every dress;
now every hat overlies every head;
now every dress overlies every pair of underpants;
now every dress overlies every foundation garment.

Section 2 - The Scenario

The Dressing Room is a room.

The player carries some capris, some jeans, a corset, a plunge bra, a thong, boy-shorts, black satin D'Orsay pumps, brown leather boots, a camisole, a cocktail dress, a bolero, a cashmere shrug, a sheer wrap, and a linen tunic.

The woolly socks are a pair of socks.
The D'Orsay pumps and the brown leather boots are pairs of shoes.
The thong and the boy-shorts are pairs of underpants.
The capris and the jeans are pairs of pants.
The tunic is a shirt.
The camisole, the corset, and the plunge bra are foundation garments.
The cocktail dress is a dress.
The bolero, the cashmere shrug, and the sheer wrap are jackets. The shrug and the wrap are transparent.

Test me with "wear capris / wear jeans / i / wear thong / i / wear dress / wear corset / wear dress / i / wear wrap / i / wear boots / wear pumps / i".

Example **Get Me to the Church on Time** *WI*

Using kinds of clothing to prevent the player from wearing several pairs of trousers at the same time.

Inform's default handling of wearable things does not make any rules about what can be worn together. Suppose, however, we have a game in which there are a large number of different garments, and we want to keep the player from wearing (say) more than one pair of pants at once:

"Get Me to the Church on Time"

A pair of pants and a shirt are kinds of thing. A pair of

pants and a shirt are usually wearable.

Some golf pants are a pair of pants. The description is "Checked in red and green, with tiny frolicking gophers every few inches."

Some tuxedo trousers are a pair of pants. The description is "Black, pressed, and slimming."

The frilly shirt is a shirt. The description of the frilly shirt is "She insisted."

The polo shirt is a shirt. The description is "Turquoise and bright yellow, the colors selected by your golfing buddies."

The player wears the golf pants and the polo shirt. The player carries the tuxedo trousers and the frilly shirt.

The Wedding Chapel Dressing Room is a room. "The bride's dressing room is a lavish suite with closets, hangers, dressmaker's dummies, boxes of straight pins and sewing notions, combs, lotions, brushes, and hair fixatives, plus room for fifteen female attendants and a photographer. Before they shoved you out of the room you even got a glimpse of a small reference library including '1001 French Braids' and 'Corset-Lacing For Beginners.'

This is the groom's dressing room. You get a framed photograph of Elvis, a dusty mirror, and the floor space of an average toilet stall."

```
The dusty mirror and the photograph of Elvis are
scenery in the Dressing Room. The description of the
mirror is "You can't really get more than a silhouette
impression of yourself." The description of Elvis is "He
reminds you that you'd better get out there before the
organist switches to Hound Dog."
```

And now the rule itself, borrowed from a later chapter:

```
Instead of wearing a pair of pants when the player is
wearing a pair of pants (called the wrong trousers):
    say "You'll have to take off [the wrong trousers] first."


Instead of wearing a shirt when the player is wearing a
shirt (called the wrong top):
    say "You'll have to take off [the wrong top] first."


When play begins:
    say "From the other side of the door, you hear the
organist move on from his instrumental interpretation of
'I Wanna Hold Your Hand' to a somewhat more spirited
rendition of 'Help! I Need Somebody!'. Okay, okay, but
you've been rushing things along since the 16th fairway,
and you can't be more than a half-hour late... Surely that
mother of hers can't blame you for that?"


Test me with "i / x trousers / wear trousers / x golf pants
/ take off golf pants / wear trousers / x frilly shirt / x polo
shirt / wear frilly shirt / doff polo shirt / wear frilly shirt".
```

If we wanted to, we could make similar kinds for hats, shoes, and so on, and have a simple but effective system of clothing. A more complicated treatment might keep track of layering and describe the player's outfit differently depending on which clothes were outermost -- an example for a later chapter.

Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

We have two things to keep track of with our layering clothing: what currently is covering something else; and what *can* cover something else. This implementation goes for a fairly simple treatment, assuming that each item of clothing will completely conceal those beneath it, and that we are not implementing entire sets of shirts, jackets, etc. But it will do for a demonstration.

```
"Bogart"
```

```
Section 1 - Clothing Behavior
```

First we make our relation to represent what *is* underneath another item:

```
Underlying relates one thing to various things. The verb
to underlie means the underlying relation. The verb to
be under implies the underlying relation.
```

And now we prevent taking a lower layer off before the thing that is worn over it:

```
Before taking off something which underlies something
(called the impediment) which is worn by the player:
say "(first removing [the impediment])[command
clarification break]";
silently try taking off the impediment;
if the noun underlies something which is worn by the
player, stop the action.
```

```
Check taking off:
    if the noun underlies something (called the
impediment) which is worn by the player, say "[The
impediment] [are] in the way." instead.
```

```
Carry out taking off:
    now the noun is not underlaid by anything.


Report taking off something:
    say "[We] [are] now wearing [a list of uppermost
things worn by the player]." instead.


Definition: a thing is uppermost if it is not under
something.
```

That covers order of clothing removal, but we also want to restrict what can be worn on top of what else. This time we need Inform to have some idea of what customarily can be layered on top of what other clothing:

```
Overlying relates one thing to various things. The verb
to overlie means the overlying relation.


Covering relates a thing (called A) to a thing (called B)
when the number of steps via the overlying relation from
A to B is greater than 0. The verb to cover means the
covering relation.
```

With these definitions, we can say that a jacket should go over a shirt and a shirt over an undershirt (say), and then Inform will know that a jacket will cover both shirt and undershirt.

```
Before wearing something when something (called the
impediment) which covers the noun is worn by the
player:
    while the player wears something which covers the
noun:
        say "(first removing [the impediment])[command
clarification break]";
        silently try taking off the impediment;
        if the player is wearing the impediment, stop the
action.
```

```
Carry out wearing:
    if the noun covers something (called the hidden item)
worn by the player, now the hidden item underlies the
noun.


Instead of looking under something which is worn by the
player:
    if something (called the underwear) underlies the
noun, say "[We] [peek] at [the underwear]. Yup, still
there.";
    otherwise say "Just [us] in there."


Instead of taking inventory:
    say "[if the player carries something][We]['re] carrying
[a list of things carried by the player][else][We]['re]
empty-handed[end if][if the player wears something].
[We] [are] wearing [a list of uppermost things worn by
the player][end if]."


To peek is a verb.
```

Notice that our inventory only describes the things that the player can see as the upper layer of clothing.

```
Section 2 - The Scenario


The Trailer is a room. "A full-length mirror is the main
amenity in here, and that suits you just fine." The full-
length mirror is scenery in the Trailer. Instead of
examining or searching the mirror, try taking inventory.


The player wears a fedora, a jacket, a shirt, some
undershorts, an undershirt, some slacks, a pair of
socks, and a pair of shoes.
```

```
The shirt underlies the jacket. The pair of socks
underlies the pair of shoes. The undershorts underlie
the slacks. The undershirt underlies the shirt.


The jacket overlies the shirt. The shoes overlie the
socks. The slacks overlie the undershorts. The shirt
overlies the undershirt.


Test me with "x mirror / remove fedora / remove jacket /
remove shirt / remove slacks / remove undershirt /
remove shoes / remove socks / remove shorts / remove
undershorts".
```

If we further wanted to prevent the player from taking off clothes in inappropriate places, we might add something like this:

```
Instead of taking off something in the presence of
someone who is not the player:
    say "[We] [are] far too modest to strip in public."
```

ExampleWidget Enterprises

Allowing the player to set a price for a widget on sale, then determining the resulting sales based on consumer demand, and the resulting profit and loss.

Suppose the player is responsible for pricing at Widget Enterprises. Widget production entails a certain fixed cost as well as a cost per unit; and somewhere out in the world there are a number of customers interested in purchasing widgets, but the player starts without knowing what this distribution looks like.

We can express the profits as an equation: the total made by selling widgets, minus the cost thereof.

The Table of Customers holds the data about customer preferences, and whenever the player selects a widget price, we consult it to determine how many customers in total would be willing to buy at that price.

"Widget Enterprises"

Widget Stand is a room.

A monetary value is a kind of value. $1.99 specifies a monetary value with parts dollars and cents.

Equation - Profit Equation
    P = nV - (F + nC)
where P is a monetary value, F is the fixed cost, C is the unit cost, V is a monetary value, and n is a number.

The fixed cost is a monetary value that varies. The fixed cost is $5.00.
The unit cost is a monetary value that varies. The unit cost is $10.66.

Table of Customers

base	maximum value
4	$26.00
5	$20.00
8	$15.00
9	$13.50
3	$9.00

To decide what number is the units sold at (V - a monetary value):
    let total units be 0;
    repeat through the Table of Customers:
        if V is less than the maximum value entry:
            increase total units by the base entry;
    decide on total units.

```
Understand "set price to [monetary value]" as setting
price to. Setting price to is an action applying to one
monetary value.


Carry out setting price to:
    let V be the monetary value understood;
    let n be the units sold at the monetary value
understood;
    let P be given by the Profit Equation;
    say "You set the price of your widgets to [V], resulting
in sales of [n] unit[s] and ";
    if P is less than $0.00:
        let L be $0.00 - P;
        say "a loss of [L].";
    otherwise if P is $0.00:
        say "break even.";
    otherwise:
        say "a profit of [P].".


Test me with "set price to $0.00 / set price to $100.00 /
set price to $15.00 / set price to $8.00 / set price to
$25.00 / set price to $14.99".
```

As written this will be a rather dull guessing game for the player; more interesting would be to enhance it into a fuller economic simulator with more control over fixed costs and customer price points.

ExampleFrozen Assets

A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping.

In our brave new world, everything will have a price, so we had better spell this out.

```
"Frozen Assets"
```

```
Price is a kind of value. $10.99 specifies a price. A thing
has a price. The price of a thing is usually $0.00. After
examining something for sale, say "It can be yours for
[the price of the noun]."
```

Now we assume a simple shopping model in which the player can't take anything without paying for it.

```
Definition: a thing is free if the price of it is $0.00.
Definition: a thing is for sale if it is not free.


Instead of taking something for sale:
    say "You'll have to pay for that."


Before buying something for sale when the money is not
in the wallet:
    say "You're broke." instead.


Before buying something for sale when the money is
free:
    say "You're broke." instead.


Before buying something for sale when the price of the
money is less than the price of the noun:
    say "Your funds do not run to the price of [the noun]."
instead.


Instead of buying something:
    decrease the price of the money by the price of the
noun;
    say "You fork over [the price of the noun] for [the
noun], leaving yourself with [the price of the money].";
    if the money is free:
        now the money is nowhere;
```

```
    now the price of the noun is $0.00;
    now the player is carrying the noun.
```

The player's money object is going to be a bit unusual, because it has value but cannot itself be bought.

```
The player carries a wallet. The wallet contains money.
The price of the money is $4.50. The printed name of
the money is "[price of the money] in cash". Understand
"cash" as the money.


Instead of taking the money:
    say "Best to leave it alone until you need to buy
something."


Instead of buying something free:
    say "[The noun] is yours already."


Instead of buying the money:
    say "The money belongs to you; you buy things with
it."
```

Now we just need something to buy.

```
The Dessert Parlor is a room. "An underlit, over-
crowded room campily furnished with a lot of gilt-frame
mirrors and twinkle lights: it is essentially a brothel of
food. The service is slow at best, and on Saturday
nights glacial. However. The wares on display more
than make up for these trivial inconveniences."


The vanilla ice cream is an edible thing in the Parlor.
The price of the ice cream is $2.45. The description is
"In the scale of ice creams, you recognize this as a very
inferior vanilla because it has no adjectives in the title."
```

The raspberry tart is an edible thing in the Parlor. The price of the tart is $4.50. The description is "An almond-laced shell packed with raspberries-under-glaze."

The syllabub is an edible thing in the Parlor. The price of the syllabub is $4.25. The description is "Whipped cream, alcohol, and lime juice, a substance without any redeeming food value whatever."

The espresso cake is an edible thing in the Parlor. The price of the espresso cake is $5.50. The description is "A lethal wedge of purest blackness."

Test me with "inventory / examine syllabub / get syllabub / buy syllabub / drop it / get it / buy raspberry tart".

Implementing caloric units for this scenario is left as an exercise for the reader.

Example **Money for Nothing**

An OFFER price FOR command, allowing the player to bargain with a flexible seller.

"Money for Nothing"

Section 1 - Prices and Bargaining

Price is a kind of value. $10.99 specifies a price with parts dollars and cents (optional, preamble optional).

A person has a price called wealth. The wealth of the player is $15.

A thing has a price called minimum value. The minimum value of a thing is usually $0.50.

A thing has a price called desired value. The desired value of a thing is usually $5.00.

Offering it for is an action applying to one price and one visible thing.

Understand "offer [price] for [something]" as offering it for.

After taking inventory, say "You have [the wealth of the player]."

Check offering it for:
    if the price understood is greater than the wealth of the player, say "You don't have that kind of cash." instead;
    if the second noun is not carried by someone, say "There's no one in a position to sell you [the second noun]." instead;
    if the second noun is carried by the player, say "[The second noun] is already yours." instead;
    if the minimum value of the second noun is greater than the price understood, say "[The holder of the second noun] cackles disdainfully. 'If yer just here to insult me you can take your business elsewhere!' he says." instead;
    if the desired value of the second noun is greater than the price understood:

```
        let difference be the desired value of the second
    noun minus the price understood;
        let difference be difference divided by two;
        decrease the desired value of the second noun by
    difference;
        now the last object offered is the second noun;
        say "'How about [desired value of the second
    noun]?' suggests [the holder of the second noun]."
    instead;
    otherwise:
        unless the desired value of the second noun is the
    price understood:
            say "From the avaricious gleam in the eye of
    [the holder of the second noun], you guess you could've
    gotten this purchase for less..."



    Carry out offering it for:
    increase the wealth of the holder of the second noun
    by the price understood;
    decrease the wealth of the player by the price
    understood;
    move the second noun to the player.



    Report offering it for:
    say "You spend [the price understood], and now you
    possess [the second noun]."



    When play begins: now right hand status line is "Your
    funds: [wealth of the player]".
```

Now, since the man does make counter-offers, it would be reasonable to let the player accept or reject those, as well:

```
    The last object offered is a thing that varies.



    Instead of saying yes when the last object offered is
    carried by a person (called seller) who is not the player:
    if the seller is not visible:
        continue the action;
```

```
    otherwise:
        now the price understood is the desired value of
    the last object offered;
        try offering the desired value of the last object
    offered for the last object offered.
```

```
Instead of saying no when the last object offered is
carried by a person (called seller) who is not the player:
    if the seller is not visible:
        continue the action;
    otherwise:
        now the last object offered is the player;
        say "You reject the offer firmly."
```

And we borrow just a line or two from a later chapter to take care of some alternate syntax the player might try:

```
Understand "offer [price] to [someone]" as a mistake
("You'll need to specify what you want to buy -- try
OFFER $1000.00 FOR BROOKLYN BRIDGE.").
Understand "offer [someone] [price]" as a mistake
("You'll need to specify what you want to buy -- try
OFFER $1000.00 FOR BROOKLYN BRIDGE.").
```

```
Understand "buy [something]" as a mistake ("You'll have
to name your price: try OFFER $1000.00 FOR
BROOKLYN BRIDGE.").
```

```
Section 2 - The Scenario
```

```
The Flea Market is a room. The crotchety man is a man
in the Market. "A crotchety man here is selling [the list of
things carried by the crotchety man]." The crotchety
man carries a broken television set, a Victorian
rhinestone brooch, and a cracked shaving mug.
```

The minimum value of the brooch is $2.50.

Test me with "offer $0.50 for mug / offer $0.50 to man / offer $6.00 for mug / offer $50.00 for brooch / offer $1.50 for brooch / offer $4.50 for brooch / no / offer $4.50 for brooch / yes".

---

269 **ExampleFabrication** *WI*

A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

When we make a new kind of value, the new named values can themselves have properties. That is convenient because, for instance, we might want to associate a material (itself the property of an object) with certain features, such as price.

"Fabrication"

Section 1 - Procedure

A material is a kind of value. The materials are silk, velvet, cotton, and wool.

Price is a kind of value. $1.99 specifies a price.

Area is a kind of value. 5 sq yards specifies an area.

Cost is a kind of value.. $1.99 per sq yard specifies a cost. A cost times an area specifies a price.

A material has a cost.

The cost of silk is usually $5.75 per sq yard. The cost of velvet is usually $9.50 per sq yard. The cost of cotton is usually $2.29 per sq yard. The cost of wool is usually $4.75 per sq yard.

A pattern is a kind of thing. A pattern has a material. A pattern has an area. A pattern has a price. The price of a pattern is usually $9.99. Understand "pattern" as a pattern. Understand "patterns" as the plural of a pattern.

After printing the name of a pattern:
    if planning:
        do nothing;
    otherwise:
        say " pattern".

To decide what price is the material price of (chosen item - pattern):
    let C be the cost of the material of the chosen item;
    let A be the area of the chosen item;
    decide on C * A.

To decide what price is the overall price of (chosen item - pattern):
    let P be the price of the chosen item;
    let M be the material price of the chosen item;
    decide on P + M.

Understand "plan [material] [pattern]" as planning it for.

Planning it for is an action applying to one material and one thing.

Carry out planning it for:
    now the material of the second noun is the material understood.

Report planning it for:
    say "You lay plans for a [material understood] [second noun], running [material price of the second noun] for materials and [price of the second noun] for the pattern itself, for a total of [overall price of the second noun]."

Section 2 - Scenario

Joanne's Fabrics is a room. Joanne's Fabrics contains a pattern bin.

The cape is a pattern. The material of the cape is velvet. The area of the cape is 9 sq yards.

The bodice is a pattern. The material of the bodice is silk. The area of the bodice is 2 sq yards. The price of the bodice is $11.99.

The cape and the bodice are in the pattern bin.

Test me with "plan silk bodice / plan velvet bodice / plan velvet cape / plan wool cape".

ExampleNickel and Dimed

A more intricate system of money, this time keeping track of the individual denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change.

Typically games which keep track of the player's wealth need only do so as an abstract number, but occasionally it becomes useful to represent money as physical coins and bills. Here is an example that does exactly that:

"Nickel and Dimed"

Section 1 - Currency

Price is a kind of value. $10.99 specifies a price with parts dollars and cents. A thing has a price. The price of a thing is usually $0.00.

Money is a kind of thing. Coin is a kind of money.

A dollar bill is a kind of money. The price of a dollar bill is $1.00. The printed name of a dollar bill is "dollar bill". Rule for printing the plural name of a dollar bill: say "dollar bills". The description of a dollar bill is "It has George Washington's head on the front, which with a bit of creative folding can be scrunched to look like a mushroom. All important things really are learned in kindergarten."

A five-dollar bill is a kind of money. The price of a five-dollar bill is $5.00. Understand "five" or "five dollar" as the five-dollar bill. The description of a five-dollar bill is "Abraham Lincoln. He looks slightly less dignified here

than he does on the penny."

A hundred-dollar bill is a kind of money. The price of the hundred-dollar bill is $100.00. Understand "hundred" or "hundred dollar" as the hundred-dollar bill. Understand "dollar" as the dollar bill. The description of a hundred-dollar bill is "It's got Benjamin Franklin, who always gets shafted: a denomination too large for anyone to carry conveniently, and a lot of local fame in Philadelphia."

Our choice of understand rules guarantees that "five dollar" will be understood as the five, but "dollar" alone as the single. We will learn more about "understand" in later chapters, but here is a test to check the functionality:

Test bills with "x hundred dollar bill / x five dollar bill / x hundred / x five / x dollar / x dollar bill".

A quarter is a kind of coin. The price of a quarter is $0.25. The description of a quarter is "One of the old-fashioned variety, not a state quarter."

A dime is a kind of coin. The price of a dime is $0.10. The description of a dime is "Franklin Roosevelt, trying not to look too annoyed that his coin is so small and thin."

A nickel is a kind of coin. The price of a nickel is $0.05. The description of a nickel is "A chubby coin, but you've always liked Thomas Jefferson, and the Monticello on the back is a nice touch."

A penny is a kind of coin. The price of a penny is $0.01. The description of the penny is "A profile of Abe Lincoln. Sometime soon they'll stop minting these, you're sure of it."

Section 2 - Ownership

Ownership relates one person (called the owner) to various things. The verb to own means the ownership relation.

Definition: a thing is owned if the player owns it.

Instead of buying something which is owned by the player:
    say "You already own [the noun]."

Instead of going somewhere when the player encloses something (called the stolen goods) which is not owned by the player:
    if the owner of the stolen goods is not a person:
        now the player owns the stolen goods;
        continue the action;
    if the owner of the stolen goods can see the player,
        say "'Hey there buddy, not so fast,' says [the owner of the stolen goods]. 'You going to buy [the stolen goods] first, or am I gonna call the cops?'";
    otherwise continue the action.

After taking inventory:
    say "Altogether, you've got [the player's cash] on your person."

To decide what price is the player's cash:
    let sum be the total price of money enclosed by the player;
    decide on sum.

```
To decide what price is the sum in (item - a container):
    let sum be the total price of the money in the item;
    decide on sum.


When play begins: now every thing carried by the player
is owned by the player.


Section 3 - Purchasing and Sales


Definition: a thing is worthless if the price of it is $0.00.
Definition: a thing is valuable if it is not worthless.


A thing can be for sale.


Rule for printing room description details of something
(called target) which is for sale (this is the disclose
prices in room description rule): say " ([price of the
target])".


Before listing contents: group money together giving
articles.


Instead of examining a for sale thing (this is the describe
things by price rule):
    say "[The noun] costs [the price of the noun], payable
to [the owner of the noun]."
```

The cashbox is a theoretical construct, not something the player will ever encounter in the course of the game. It contains all the money that is available for non-player characters to use in making change. If we wanted, we could give each character his own stash of change, but this would increase the likelihood that any given person would run out of cash to make change with. (And in this example there is only one vendor

anyway.)

The cashbox is a container. The cashbox contains 10 pennies. The cashbox contains 10 nickels. The cashbox contains 10 dimes. The cashbox contains 10 quarters. The cashbox contains 10 dollar bills. The cashbox contains 10 five-dollar bills.

The block buying rule is not listed in the check buying rules.

```
Check buying something:
    if the noun is not for sale, say "[The owner of the
noun] does not want to sell you [the noun]." instead;
    if the player's cash is less than the price of the noun,
say "You can't afford the asking price of [the price of the
noun] for [the noun]." instead.
```

```
Carry out buying something:
    let sum paid be $0.00;
    while sum paid is less than the price of the noun:
        let current target be the price of the noun minus the
sum paid;
        let bill offered be the best money from the player
for the current target;
        if the bill offered is money:
            move the bill offered to the owner of the noun;
            now the bill offered is spent;
            increase the sum paid by the price of the bill
offered;
            let current target be the price of the noun minus
the sum paid;
    say "You hand [the owner of the noun] [a list of spent
money]. [run paragraph on]";
    let change be $0.00;
    if the sum paid is greater than the price of the noun:
        now the change is the sum paid minus the price of
the noun;
    if change is greater than the sum in the cashbox:
        now the player carries every spent money;
        now every spent thing is fresh;
        say "'Whoa,' says [the owner of the noun], handing
```

```
the cash back to you. 'I can't make change for that,
man, sorry.'" instead;
    now every spent thing is in the cashbox;
    now every spent thing is fresh;
    while change is greater than $0.00:
        let change bill be the best money from the cashbox
for change;
        decrease change by the price of the change bill;
        now change bill is spent;
        move change bill to player;
    if money is spent, say "[The owner of the noun]
makes change with [a list of spent money]. [run
paragraph on]";
    now every spent thing is fresh;
    if the noun is not enclosed by the player and the
owner of the noun can touch the noun:
        say "'Here ya go,' says [the owner of the noun],
handing [the noun] to you. [run paragraph on]";
        move the noun to the player;
    now the player owns the noun.



Money can be spent or fresh.



Report buying something:
    if the player owns the noun,
        say "Your transaction is now complete, leaving you
with [the player's cash]."
```

We've skipped over defining what makes a denomination the best for a given transaction, so we'd better do that now. Our goal is to avoid ever having the player gratuitously overpay -- he should always offer the smallest amount of money that will meet the price of what he's buying.

We also assume that all money "enclosed by the buyer" -- that is, somewhere in the buyer's possession -- is available for use. This might not be true in a game where the player could pick up, say, a sealed lucite container with a ten-dollar bill inside; in that case we would have to define our terms more rigorously, perhaps by requiring that the bills be both enclosed and touchable by the buyer. The touchability check adds an extra layer of calculation, however, and since it is not necessary in this example (and probably not in most other cases either), we'll leave it out:

```
Definition: money is costly if its price is $2.50 or more.
Definition: money is cheap if its price is $0.99 or less.
```

```
Functional relation is a kind of value. The functional
relations are overpayment, underpayment and
irrelevant. Money has a functional relation.
```

```
To decide what money is the best money from (buyer -
a thing) for (cost - a price):
    repeat with bill offered running through money:
        if the bill offered is enclosed by the buyer:
            if the price of the bill offered is the cost, decide
on the bill offered;
            if the price of the bill offered is greater than the
cost, now the functional relation of bill offered is
overpayment;
            otherwise now the functional relation of the bill
offered is underpayment;
        otherwise:
            now the functional relation of the bill offered is
irrelevant;
    [say "underpayment: [a list of underpayment money]
    overpayment: [a list of overpayment money]";]
    if the total price of underpayment money is less than
the cost:
        decide on the cheapest money which is
overpayment;
    otherwise:
        decide on the costliest money which is
underpayment.
```

Notice the "say underpayment/overpayment section..." noted out, above. This is for debugging purposes: when writing complex code, it is sometimes useful to put in lines that will say explicitly what is going on. We can enclose them in brackets and Inform will ignore them as though they were comments; if we run into any problems with the code later, we can erase the brackets and see the diagnostic printed to the screen as we play.

```
Instead of giving money to someone:
    say "Best to keep the transaction simple by buying
whatever you want."
```

Section 4 - The Scenario

The player carries 2 dollar bills. The player carries a nickel. The player carries 2 pennies. The player carries a five-dollar bill. The player carries 1 hundred-dollar bill.

The Subway Station is a room.

The Bitterly Cold Street is north of the Subway Station. "Even though there is no actual snow or ice, the street is about as cold as you can stand, for which reason walking the twenty blocks uptown is not an acceptable option." The Bitterly Cold Street contains a dollar bill.

The newspaper man is a man in the Subway Station. "A newspaper man in a knit cap and fingerless gloves is hopping up and down behind his stand[if the turn count is 1]. Cold weather, caffeine overdose, or mental illness? You may never know. Welcome to New York[end if]." The description is "Eye contact with strangers is something to avoid around here."

The stand is a supporter in the Station. The stand is scenery.

A copy of the New York Times is on the stand. The price of the New York Times is $1.25.

A pack of gum is on the stand. The price of the gum is $0.40.

A paperback novel is on the stand. The price of the paperback novel is $7.99.

A packet of trading cards is on the stand. The price of the packet is $0.99.

When play begins:
    now every thing on the stand is owned by the newspaper man;
    now every thing on the stand is for sale.

We could have done all that by hand, but the initialization requires a little less work.

The ticket machine is a container in the Subway Station. It is fixed in place. The description of the ticket machine is "An LED screen on the front instructs you to insert [remaining ticket total] to complete your purchase. You also notice that the NO CHANGE light is lit up." The light is part of the ticket machine. The printed name of the light is "no change light". Understand "no change" or "no change light" as the light.

The description of the light is "In the whole of your recollection, the ticket machine has actually had change a total of twice. Usually, as now, the no-change light gleams angrily, daring you to put in more than you owe." A cash return button is part of the ticket machine. Instead of pushing the cash return button: say "The ticket machine regurgitates [the list of things in the ticket machine]."; now every thing in the ticket machine is carried by the player. Instead of taking something which is in the ticket machine: say "The ticket machine has swallowed your money, but it can be retrieved (you hope) with the cash return button."

Instead of inserting a hundred-dollar bill into the ticket

```
machine:
    say "What, are you nuts?"


To decide what price is the remaining ticket total:
    let absolute cost be $2.25;
    let remaining cost be absolute cost minus the total
price of things in the ticket machine;
    if remaining cost is less than $0.00, decide on $0.00;
    decide on remaining cost.


Instead of inserting something which is not money into
the ticket machine: say "The ticket machine only
accepts money, not other tokens of your esteem and
regard."


Instead of inserting a penny into the ticket machine:
    say "The penny rattles out again mockingly: not even
the ticket machine thinks these are worth anything."


A subway pass is a kind of thing. 15 subway passes are
in the cashbox. The description of a subway pass is
usually "A rectangle of thick lavender paper with a black
magnetic stripe running up the back side. It is good for
one trip on the subway."


After inserting something into the ticket machine:
    if the remaining ticket total is $0.00:
        let purchased ticket be a random subway pass in
the cashbox;
        if purchased ticket is not a subway pass, say "The
ticket machine grunts disobligingly and then the
unwelcome word MALFUNCTION parades across the
LED screen, three letters at a time." instead;
        repeat with item running through things in the
machine:
            now the item is nowhere;
        move purchased ticket to player;
        say "The ticket machine beeps obligingly and
disgorges a single subway pass.";
```

```
    otherwise:
        say "The ticket machine beeps obligingly and
    adjusts its price down to [remaining ticket total]."
```

And because even though the ticket machine is a container, we don't want to say (empty) after it in the room description:

```
Rule for printing room description details of the ticket
machine:
    do nothing instead.
```

```
Test me with "buy novel / n / get dollar / s / buy novel".
```

After all that, we should probably give the player a chance to win, as well:

```
The turnstile is south of the Subway Station. "A turnstile
is all that separates you from the subway platform
stairs." The turnstile is north of the Platform. The
turnstile is a door. Before going down in the presence of
the turnstile, try going south instead. The turnstile is
openable. The turnstile is open.
```

```
Instead of going through the turnstile when the player
carries a subway pass: say "You enter the turnstile and
begin your journey uptown..."; end the story finally
saying "At last". Instead of going through the turnstile:
say "You can't go through the turnstile without a subway
pass. They're very strict about this."
```

```
Instead of inserting money into the turnstile: say "The
turnstile takes passes, not money." Instead of inserting
a subway pass into the turnstile, try entering the
turnstile.
```

```
Test more with "buy times / put all but five-dollar bill in
machine / press button / buy gum / buy cards / i / put
dollar in machine / g / put quarter in machine / i / d".
```

In fairness to the Metropolitan Transit Authority, we should admit that most of the ticketing machines in the real New York subway are better than this, and will accept, say, a debit card. But that would be so much less exciting to implement.

---

283 **Example** **Introduction to Juggling** *WI*

Assortment of equipment defined with price and description, in a table.

Suppose we have a whole catalog-full of equipment that the player might want to purchase and use. We'll start by defining our purchasing rules:

```
"Introduction to Juggling"
```

We allow things to have prices, and the player's price to indicate how much money the player has:

```
Section 1 - Mail-ordering defined

Price is a kind of value. $100.99 specifies a price.

The player has a price. The price of the player is
$60.00.
```

Because we're allowing the player to order things that he can't currently see, we need to borrow a special kind of grammar from the Understanding chapter. All our orderable items in this example are toys, so "any toy" means any object of the toy kind, whether or not it is in view at the moment:

```
Understand "buy [any toy]" as ordering. Understand the
command "order" as something new. Understand the
command "order" as "buy".
```

```
Ordering is an action applying to one visible thing.


Check ordering:
    if the cost of the noun is greater than the price of the
player, say "You only have [price of the player], while
[the noun] would cost [cost of the noun]." instead.


Carry out ordering:
    move the noun to the player;
    decrease the price of the player by the cost of the
noun.


Report ordering:
    say "You order [a noun], which is delivered instantly."
```

We should also handle the situation where the player orders another of something he has already bought and which is right in front of him:

```
Instead of buying something:
    say "You already have [a noun]."
```

So much for the general rules for this scenario. Now we move on to particulars: the actual items the player is allowed to order. Each item will have a description, a price, and a difficulty representing how skilled the player must be in order to make use of that item.

Since we are going to use price and difficulty in the table that defines our juggling equipment, we need to mention these kinds of value before the line that says how toys are defined.

```
Section 2 - The Scenario


Difficulty is a kind of value. The difficulties are easy,
moderate, hard. The player has a difficulty. The difficulty
```

of the player is easy.

The plural of toy is toys. A toy is a kind of thing. Some toys are defined by the Table of Juggling Equipment.

Table of Juggling Equipment

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

Notice that we are allowed to define "description" and other already-known properties in the table as well.

Backstage is a room. "A muffled black room with felt on the floors and walls. A glowing sign over the stage door says SHOW IN PROGRESS."

The Juggling Equipment Catalog is a thing in Backstage.

Instead of examining the Catalog:
    say "You read through the offerings, including: [paragraph break]";
    repeat through Table of Juggling Equipment:
        say "[bold type][toy entry][roman type]: [description entry] [cost entry], [restriction entry]. [paragraph break]".

```
When play begins:
    now right hand status line is "Budget: [price of the
player]";
    now left hand status line is "[location], feeling [if the
difficulty of player is easy]incompetent[end if][if the
difficulty of player is moderate]moderately skilled[end if]
[if the difficulty of player is hard]highly skilled[end if]".
```

And of course this will be no fun unless the player is allowed to use the equipment:

```
Understand "juggle [something]" as juggling.


Juggling is an action applying to one thing.


Check juggling:
    if the noun is not a toy listed in the Table of Juggling
Equipment, say "You can't juggle [a noun]!" instead;
    if the difficulty of the noun is greater than the difficulty
of the player, say "You're not quite ready to juggle
something like [the noun]. Better to start with an easier
toy." instead.


Carry out juggling:
    if a random chance of 1 in 3 succeeds:
        if the difficulty of the player is less than hard and
the difficulty of the player is the difficulty of the noun:
            now the difficulty of the player is the difficulty
after the difficulty of the player.


Report juggling:
    say "[outcome of the noun][paragraph break]".


Instead of burning the fireball set:
```

say "It will flame by itself when you use it."

Test me with "read catalog / buy economy / buy beanbag / juggle economy / juggle beanbag / juggle beanbag / juggle beanbag / juggle beanbag / juggle beanbag / juggle beanbag / juggle economy / juggle economy / juggle economy / buy fireball set / juggle fireball".

---

129 **Example** **Do Pass Go** *WI*

A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

"Do Pass Go"

Go is a room. "A giant square area, where you and your other pewter ornament friends gather before setting out to purchase London."

The pair of dice is carried by the player.

The pair of dice has a number called first die. The pair of dice has a number called second die. The first die of the pair is 6. The second die of the pair is 6. Rule for printing the name of the pair of dice while taking inventory: say "pair of dice".
Rule for printing the name of the pair of dice: say "pair of dice showing [first die of the pair plus second die of the pair]".

To say detailed state of the dice:

```
    if the first die of the pair is the second die of the pair,
say "double [first die of the pair]";
    otherwise say "[first die of the pair] and [second die of
the pair]".


The description of the pair of dice is "The pair of dice
are [if the dice are carried]itching to be
rolled[otherwise]showing [detailed state of the dice][end
if]."


Rolling is an action applying to one carried thing.
Understand "roll [something preferably held]" as rolling.


Check rolling when the noun is not the pair of dice: say
"Not something you can roll." instead.
Carry out rolling:
    now the pair of dice is in the holder of the actor;
    now the first die of the pair of dice is a random
number from 1 to 6;
    now the second die of the pair of dice is a random
number from 1 to 6.
Report rolling:
    say "You roll [detailed state of the dice]."


Test me with "i / roll dice / look / x dice / get dice / x dice
/ roll dice / roll dice / roll dice / roll dice / roll dice / roll
dice / roll dice".
```

Because we remember the states of the individual dice, not just a total, we can make use of the combination rolled.

```
The doubles count is a number that varies.
After rolling:
    if the first die of the pair is the second die of the pair,
increment the doubles count;
    otherwise now the doubles count is 0;
    continue the action.
```

Jail is a room. "This is Jail, and not the Just Visiting periphery, either."

Every turn when the doubles count is 3:
    say "The blue-uniformed policemen blows his whistle and beckons you sternly...";
    now the player carries the pair of dice;
    now the player is in Jail;
    now the doubles count is 0.

Every turn when the doubles count is 1 and the player is in Jail:
    say "The warden gruffly releases you.";
    now the player carries the pair of dice;
    now the player is in Go.

Example **Wonka's Revenge**

A lottery drum which redistributes the tickets inside whenever the player spins it.

"Wonka's Revenge"

The Caribou Lodge is a room. "Hundreds of expectant faces are turned your way from every table." A lottery drum is in the Lodge. "Before you is the lottery drum[if we have spun the drum], ready to disgorge a ticket[otherwise], waiting to be spun[end if]." In the drum are a red ticket, an orange ticket, a yellow ticket, a green ticket, a blue ticket, a purple ticket, and a ticket of pure gold. The drum is closed and openable.

Understand "spin [something]" as spinning.

```
Spinning is an action applying to one thing.


Check spinning: if the noun is an open container which
contains something, say "[The list of things in the noun]
would fly out." instead.


Carry out spinning a container:
    shuffle the contents of the noun.


Report spinning:
    if the noun contains something, say "You rattle [if the
noun is transparent][the list of things in the noun]
[otherwise]the stuff[end if] in [the noun].";
    otherwise say "Nothing results of your shaking [the
noun]."
```

Inform keeps track of the order in which things have been put into a container. If we want to change that order without the player's intervention, we can move the things ourselves.

```
To shuffle the contents of (basket - a container):
    let moves be the number of things in the basket;
    repeat with counter running from 1 to moves:
        move a random thing in the basket to the basket.


After opening the drum when we have spun the drum for
the first time:
    if something (called the pick) is in the drum:
        try searching the drum;
        say "[The pick] it is, then.";
        silently try taking the pick;
        if the pick is the ticket of pure gold, end the story
finally;
        otherwise end the story saying "Oh well, better luck
next time."
```

Test me with "open drum / look in drum / close drum / spin drum / open drum".

 Example **Jokers Wild**

A deck of cards which can be shuffled and dealt from.

Suppose we want a deck of cards which the player can shuffle and draw from. Our first (rather tedious) task is merely to set up the deck as a table:

"Jokers Wild"

Suit is a kind of value. The suits are hearts, clubs, diamonds, and spades.

Table of Cards

suit	value
diamonds	1
diamonds	2
diamonds	3
diamonds	4
diamonds	5
diamonds	6
diamonds	7
diamonds	8
diamonds	9
diamonds	10
diamonds	11
diamonds	12
diamonds	13
spades	1
spades	2
spades	3
spades	4
spades	5
spades	6
spades	7

8 spades
9 spades
10 spades
11 spades
12 spades
13 spades
1 hearts
2 hearts
3 hearts
4 hearts
5 hearts
6 hearts
7 hearts
8 hearts
9 hearts
10 hearts
11 hearts
12 hearts
13 hearts
1 clubs
2 clubs
3 clubs
4 clubs
5 clubs
6 clubs
7 clubs
8 clubs
9 clubs
10 clubs
11 clubs
12 clubs
13 clubs

We're going to describe the higher numbers as face cards, so it helps to write a new "to say" phrase.

```
To say (count - a number) as a card value:
    choose row with a value of count in the Table of
Value Names;
    say "[term entry]".


Table of Value Names

term
value
"1ace"
```

2 "deuce"
3 "three"
4 "four"
5 "five"
6 "six"
7 "seven"
8 "eight"
9 "nine"
10 "ten"
11 "jack"
12 "queen"
13 "king"

Now we get the shuffling of the deck from "sort in random order", so:

```
Understand "shuffle" as shuffling. Shuffling is an action
applying to nothing.

Carry out shuffling:
    sort the Table of Cards in random order;
    say "You expertly rearrange the cards.".

When play begins:
    sort the Table of Cards in random order.
```

This will continue to work properly even as the deck is partially depleted. Speaking of which, suppose we want the player to be able to toss the cards one-by-one into a hat. They are going to need to be removed from the deck, so:

```
Understand "toss" or "toss a card" or "toss card" as
tossing.

Tossing is an action applying to nothing.
```

```
Check tossing:
    if the number of filled rows in the Table of Cards is 0,
say "The deck is empty." instead.


Carry out tossing:
    repeat through the Table of Cards:
        let new value be value entry;
        let new suit be suit entry;
        say "You throw the [value entry as a card value] of
[suit entry] at the top hat, and [if a random chance of 1
in 3 succeeds]hit[otherwise]miss[end if].";
        blank out the whole row;
        rule succeeds.
```

If we wanted to simulate a slightly more stimulating game, we could instead have a second table to represent the player's hand of cards and record each card drawn. That would get long for the purposes of example, however, so instead we will just admit that the player's life is an empty husk of existence:

```
The Empty Room is a room. "It has come to this: sitting
on the bare floor of Lulu's apartment with nothing to
amuse you but a deck of cards and the top hat from last
year's act. You reckon [the number of filled rows in the
Table of Cards in words] cardtosses are all that stand
between you and the utter pointlessness of existence.

Once again you curse Lulu for running off with that
joker."

The player is carrying the deck of cards. The top hat is
an open container in the Empty Room. It is scenery.

Test me with "toss / again / again / again / again / again
/ again / again".
```

A deck of cards with fully implemented individual cards, which can be separately drawn and discarded, and referred to by name.

We've simulated a deck of cards before, but only as entries in a table. This time we're going to do it more completely, with card objects that can be drawn and discarded, and referred to by name. The tedious way to do this would be to make 52 objects by hand and laboriously write out their names and understand rules.

A more sensible way is to make 52 identical card objects, assign them ranks and suits, and allow Inform to generate and parse their names automatically.

So:

```
"Tilt"

Section 1 - Cards

Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades. Understand "heart" as hearts.
Understand "club" as clubs. Understand "diamond" as
diamonds. Understand "spade" as spades. [Providing
the singular forms means that Inform will also
understand >EXAMINE SPADE, >DISCARD CLUB, and
so on.]

A card is a kind of thing. A card has a suit. A card has a
number called rank. Understand the suit property as
describing a card. Understand the rank property as
describing a card.

52 cards are in the card repository.
```

Now, we're going to describe the higher numbers as face cards, so it

helps to write a new "to say" phrase, just as we did in Jokers Wild. (A subsequent version of this example shows how to print card values with red and black symbols representing the different suits; see "Tilt 3".)

```
To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".


Rule for printing the name of a card (called target):
    say "[rank of the target as a card value] of [suit of the
target]"


Table of Value Names

term entry
"ace/A/one"
"deuce/two"
"three"
"four"
"five"
"six"
"seven"
"eight"
"nine"
"ten"
"jack/knave/J"
"queen/Q"
"king/K"
```

This is enough already to let inform understand things like "ten clubs", but we want to add a couple of refinements. For one thing, we'd like to accept "of" when it appears in phrases such as "ten of clubs" (but not generically otherwise); for another, we'd like the player to be able to use various names for ranks. To this end, we need to borrow from the Activities chapter and modify the player's command before attempting to understand it:

```
After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
```

```
            cut the matched text;
        repeat through the Table of Value Names:
            while the player's command includes topic entry:
                replace the matched text with value entry.
        [This allows Inform to understand "ace", "deuce",
    "king", etc., as numerical ranks.]
```

It may be a bit confusing that the Table of Value Names has both a topic column and a term column, to all appearances essentially identical. But items in the topic column can be matched against the player's input, whereas items in other kinds of text column can be printed out; the two kinds of text are not treated identically by Inform, so we need to have both. Notice that the topic column contains entries like "jack/knave," which will match either "jack" or "knave" in the player's input.

Now to set up the deck at the outset. With some intelligent looping, we avoid having to declare every combination of suit and number individually:

```
When play begins:
    reconstitute deck.


To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;
            now item is in the deck of cards;
        now current suit is the suit after the current suit.
```

And now we need a simple setting and some actions to manipulate the deck with:

```
Section 2 - The Deck and the Discard Pile


The Empty Room is a room. "Nothing to see here."
```

```
The deck of cards is in the Empty Room. It is a closed
unopenable container. The description is "A standard
poker deck."


The discard pile is a closed unopenable container. The
description is "Cards in this game are discarded face-
down, so the discard pile is not very interesting to see.
All you can observe is that it currently contains [if the
number of cards which are in the discard pile is less
than ten][the number of cards which are in the discard
pile in words][otherwise]about [the rounded number of
cards which are in the discard pile in words][end if]
card[s]."


To decide what number is the rounded number of
(described set - a description of objects):
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.
```

The above phrase rounds a number to the nearest five, so that the player is not autistically able to count a large number of cards in the discard pile at a single glance.

This next bit is an optional borrowing from the Activities chapter: we want to prevent Inform printing things like "You can see a discard pile (closed) here.", since we don't want the player to think of the piles as containers, even though Inform thinks of them in those terms.

```
Rule for printing room description details of something:
do nothing instead.
```

Finally, we want the player to use "draw" and "discard" to manipulate his hand of cards:

```
Section 3 - Drawing and Discarding Actions


Understand the commands "take" and "carry" and "hold"
```

and "get" and "drop" and "throw" and "discard" as something new.

Understand "take [text]" or "get [text]" or "drop [text]" as a mistake ("Here, you only draw and discard. Nothing else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as drawing. Drawing is an action applying to nothing. The drawing action has an object called the card drawn.

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

Understand "discard [card]" as discarding. Discarding is

an action applying to one thing.

Check discarding:
    if the player does not carry the noun, say "You can only discard cards from your own hand." instead.

Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile to the location.

Report discarding:
    say "You toss [the noun] nonchalantly onto the discard pile."

Seeding is an action out of world. Understand "seed" as seeding. Carry out seeding: seed the random-number generator with 5681.

Test me with "seed / draw / g / g / g / g / i / discard seven of spades / draw / discard six / draw / i / discard hearts / discard six of diamonds card / draw / draw / i / discard spades card / draw / discard king card".

ExampleTilt 2

A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

In our previous implementations of playing cards, we've gotten as far as creating decks of individual cards that the player can draw and discard. But in a poker game, one doesn't just have a collection of cards: one has a hand of a specific kind.

Here we take on the job of writing an inventory listing for a poker hand

that will reflect the real value of what the player has drawn. To do this, we create a rulebook to sort and assess the cards in the player's hand; its possible return values are limited to the kinds of poker hands that exist, from "high card" to "royal flush".

The first three sections, creating the deck of cards and the means to parse their names, are identical to those we've already seen in Tilt 1; new material begins at section 4.

For the purposes of demonstration, we're simulating something akin to five-card draw without wilds; stud or hold-em variations would add some other complexities.

```
"Tilt"


Section 1 - Cards


Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades. Understand "heart" as hearts.
Understand "club" as clubs. Understand "diamond" as
diamonds. Understand "spade" as spades.


A card is a kind of thing. A card has a suit. A card has a
number called rank. Understand the suit property as
describing a card. Understand the rank property as
describing a card.


52 cards are in the card repository.


To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".


Rule for printing the name of a card (called target):
```

```
    say "[rank of the target as a card value] of [suit of the
target]"



Table of Value Names

topic                    value
"ace"                    "A/one"
"deuce"                  "two"
"three"                  "3"
"four"                   "4"
"five"                   "5"
"six"                    "6"
"seven"                  "7"
"eight"                  "8"
"nine"                   "9"
"ten"                    "10"
"jack"                   "J/knave"
"queen"                  "Q"
"king"                   "K"



After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
            cut the matched text;
    repeat through the Table of Value Names:
        while the player's command includes topic entry:
            replace the matched text with value entry.



When play begins:
    reconstitute deck.



To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;
            now item is in the deck of cards;
```

now current suit is the suit after the current suit.

Section 2 - The Deck and the Discard Pile

The Empty Room is a room. "Nothing to see here."

The deck of cards is in the Empty Room. It is a closed unopenable container. The description is "A standard poker deck."

The discard pile is a closed unopenable container. The description is "Cards in this game are discarded face-down, so the discard pile is not very interesting to see. All you can observe is that it currently contains [if the number of cards which are in the discard pile is less than ten][the number of cards which are in the discard pile in words][otherwise]about [the rounded number of cards which are in the discard pile in words][end if] card[s]."

To decide what number is the rounded number of (described set - a description of objects):
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.

Rule for printing room description details of something: do nothing instead.

Section 3 - Drawing and Discarding Actions

Understand the commands "take" and "carry" and "hold" and "get" and "drop" and "throw" and "discard" as something new.

Understand "take [text]" or "get [text]" or "drop [text]" as a mistake ("Here, you only draw and discard. Nothing else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as drawing. Drawing is an action applying to nothing. The drawing action has an object called the card drawn.

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

```
Understand "discard [card]" as discarding. Discarding is
an action applying to one thing.


Check discarding:
    if the player does not carry the noun, say "You can
only discard cards from your own hand." instead.


Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile
to the location.


Report discarding:
    say "You toss [the noun] nonchalantly onto the
discard pile."
```

New material begins here. We want to start by grouping cards together, but identifying poker hands only if the player holds a full five cards.

```
Section 4 - Assessing Hands


Before listing contents while taking inventory: group
cards together.


Before grouping together cards:
    if the number of cards carried by the player is 5:
        say "[run paragraph on]";
        follow the hand-ranking rules;
        if the rule succeeded, say "[the outcome of the
rulebook]";
        otherwise say "some random cards";
        if the outcome of the rulebook is pair outcome, say
" of [rank of the first thing held by the player as a card
value]s";
    otherwise:
        say "[number of cards carried by the player in
```

```
words] assorted cards";
    say " (".



Rule for grouping together cards:
    say "[list hand]".



To say list hand:
    let chosen card be the first thing held by the player;
    while chosen card is a card:
        say "[chosen card]";
        now chosen card is the next thing held after
chosen card;
        if chosen card is a card, say ", ".



After grouping together cards:
    say ")".
```

The ranking of poker hands traditionally depends on three features: 1) whether all the cards are of the same suit (flush); 2) whether the cards constitute a numerical run of ranks (straight); and 3) how many cards or sets of cards are of matching rank (pairs, three of a kind, and four of a kind). Here we will start by assessing our hand to determine these qualities:

```
The hand-ranking rules is a rulebook. The hand-ranking
rules have outcomes royal flush, straight flush, four of a
kind, full house, flush, straight, three of a kind, two pairs,
pair, high card.



The hand-ranking rulebook has a truth state called the
flushness.
The hand-ranking rulebook has a truth state called the
straightness.



The hand-ranking rulebook has a number called the pair
count.
```

```
The hand-ranking rulebook has a number called the
triple count.
The hand-ranking rulebook has a number called the
quadruple count.
```

For convenience in identifying hand features, and for elegance when we print the hand-listing, we start by sorting the cards in the player's hand so that the high-ranked cards are listed first. It is rare that we want to concern ourselves with this, but as we saw in the section on "Looking at containment by hand" in the chapter on Change, Inform keeps an ordered list of the items inside any given container; so it does order the objects in the player's hand, and the ordering depends on which things were added to the hand most recently. By moving something to the player's hand again (even if it was already there), we change this ordering, and wind up with a sorted hand.

```
A card can be sorted or unsorted. A card is usually
unsorted.
```

```
Definition: a card is high if its rank is 11 or more.
Definition: a card is low if its rank is 4 or less.
```

```
A hand-ranking rule (this is the initial sort rule):
    now every card is unsorted;
    while the player carries an unsorted card:
        let item be the lowest unsorted card held by the
player;
        move item to the player;
        now the item is sorted;
    if sort-debugging is true, say "-- after initial sort: [list
hand]".
```

This last printing instruction is there for diagnostic purposes: later we'll add a testing command to turn debugging on and off; when it's on, the game will print out its card list at various stages in sorting, to help us trouble-shoot any problems. In normal play, however, this will be off.

Next up, a check to see whether the player has a flush:

```
A hand-ranking rule (this is the finding flushness rule):
    let called suit be the suit of a random card carried by
```

```
the player;
    if every card carried by the player is called suit, now
flushness is true.
```

Now we check for straights; this is slightly complicated by the fact that an ace can be either the bottom of a low straight (lower than 2) or the top of a high straight (higher than king), so we explicitly check both possibilities.

```
A hand-ranking rule (this is the finding straightness
rule):
    now straightness is true;
    let N be the rank of the highest card which is carried
by the player;
    repeat with current rank running from N - 4 to N:
        now the test rank is the current rank;
        unless the player carries a matching card:
            if the current rank is N - 4 and the current rank is
9 and the player carries an ace card, do nothing; [this
covers the case where an ace could be the top card of
the sequence]
            otherwise now straightness is false.
```

And finally, we need to identify any groups of cards of the same rank. We want to know how many groups there are and how large each group is (though in practice there can only be one group of three or four in a standard-sized poker hand). We also want to mark any grouped cards so that we can move them to the front of the player's hand when we take inventory.

```
A card can be quadrupled, tripled, paired or
uncombined.
```

```
Test rank is a number that varies. Definition: a card is
matching if its rank is the test rank.
```

This definition is a convenience so that we don't have to write so many explicit loops in the following rule:

```
A hand-ranking rule (this is the counting multiples rule):
    now every card is uncombined;
    repeat with current rank running from 1 to 13:
```

```
        now test rank is current rank;
        let N be the number of matching cards held by the
    player;
        if N is 4:
            increment the quadruple count;
            now every matching card held by the player is
    quadrupled;
        if N is 3:
            increment the triple count;
            now every matching card held by the player is
    tripled;
        if N is 2:
            increment the pair count;
            now every matching card held by the player is
    paired.
```

Next we tweak our sorting to reflect the make-up of the hand. There are two reasons why this might differ from the straight highest-to-lowest sort we did earlier:

1) we want to list aces as high unless they are serving as the bottom of a low straight, in which case they should appear last;

2) we want combinations to appear at the front of the list, sorted from highest value to lowest value: larger combinations first, then smaller combinations, and combinations of equal size sorted by rank.

```
A hand-ranking rule (this is the move aces up unless
there's a low straight rule):
    unless the straightness is true and the lowest card
carried by the player is an ace card and the rank of the
highest card carried by the player is 5,
        now every ace card which is carried by the player
is carried by the player;
    if sort-debugging is true, say "-- after ace movement
rule: [list hand]".
```

```
A hand-ranking rule (this is the move pairs forward rule):
    while the player carries a paired card:
        let selection be the lowest paired card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after pairs movement:
[list hand]".
```

```
A hand-ranking rule (this is the raise ace pairs rule):
    if the player carries exactly two ace cards:
        repeat with item running through ace cards which
are carried by the player:
            move item to the player;
    if sort-debugging is true, say "-- after paired-ace
movement: [list hand]".


A hand-ranking rule (this is the move multiples forward
rule):
    while the player carries a tripled card:
        let selection be the lowest tripled card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    while the player carries a quadrupled card:
        let selection be the lowest quadrupled card which
is carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after multiples
movement rule: [list hand]".


Definition: a card is ace if its rank is 1.
Definition: a card is king if its rank is 13.
```

Now, having determined the salient qualities of our hand, we run through rules in order from the highest kind of poker combination to the lowest. Because of the order of the source, Inform will choose whichever combination applies first.

```
A hand-ranking rule (this is the royal-flush rule):
    if flushness is true and straightness is true and the
highest card carried by the player is king and the lowest
card carried by the player is ace, royal flush.


A hand-ranking rule (this is the straight-flushes rule):
    if flushness is true and straightness is true, straight
flush.
```

```
A hand-ranking rule (this is the four-of-a-kind rule):
    if the quadruple count is 1, four of a kind.

A hand-ranking rule (this is the full-house rule):
    if the pair count is 1 and the triple count is 1, full
house.

A hand-ranking rule (this is the flushes rule):
    if flushness is true, flush.

A hand-ranking rule (this is the straights rule):
    if straightness is true, straight.

A hand-ranking rule (this is the three-of-a-kind rule):
    if triple count is 1, three of a kind.

A hand-ranking rule (this is the two-pair rule):
    if the pair count is 2, two pairs.

A hand-ranking rule (this is the pair rule):
    if the pair count is 1, pair.

A hand-ranking rule (this is the default rule):
    high card.
```

And finally, we need to define our debugging variable here, even though we won't give the player the ability to turn it on and off except in the special testing section.

```
Sort-debugging is a truth state that varies.
```

For many examples, a test-me script is enough to prove that the example does what it ought. This example, though, is a bit more complicated, and hard to test randomly. The remainder of the source here shows how we might write a test to verify the desired behavior of our rulebook. Those who are only interested in the rulebook itself can stop reading at this point.

```
Section 5 - Testing hand identification - Not for release
```

For the sake of testing our rules, we provide an apparatus that will load the player's hand up with sample hands of each kind, then show the result to make sure that the hand is being correctly identified.

```
Understand "debug sorting" as debugging hand sorting.
Debugging hand sorting is an action out of world.

Carry out debugging hand sorting:
    if sort-debugging is false, now sort-debugging is true;
    otherwise now sort-debugging is false.

Report debugging hand sorting:
    say "Sort debugging is now [if sort-debugging is
true]on[otherwise]off[end if]."

Test me with "draw / g / g / g / g / force hand / g / g / g /
g / g / g / g / g / g / g / g / g".
```

The somewhat rough-and-ready principle of this table is that we will overwrite the cards in the player's hand by resetting their ranks and suits; every five rows of the table represent a new poker hand for the game to attempt to sort and identify. These include one example of each of the major kinds of poker hand, plus a couple of variations involving aces which test the special sorting rules.

Table of Testing Hands

set card
[royal flush]
s1
s13
s12
s11
s10
[straight flush]
c12
c11
c10
c9
c8
[four of a kind]
h7
s7
d7
c7
[full house]
s1
h1
s10
c10
[flush]
h3
h5
h7
h11
h12
[straight]
s9
s13
d12
c11
h10
[three of a kind]
s8
d8
c8
s4
s3
[two pairs]
s6
c6
d7
h7
h9
[two pairs, ace high]
s6
c6
d7
h7
h1
[pair]
s12
d12

```
3 spades
4 clubs
[10 high]
[jack high]
king hearts
9 spades
7 clubs
6 diamonds
[tricky sorting: low straight]
2 diamonds
3 spades
4 diamonds
5 diamonds
```

Understand "force hand" as forcing a hand. Forcing a hand is an action out of world.

Current marker is a number that varies.

Carry out forcing a hand:
    repeat with item running through cards which are carried by the player:
        increment current marker;
        if current marker is greater than the number of filled rows in the Table of Testing Hands, now current marker is 1;
        choose row current marker in the Table of Testing Hands;
        now the suit of item is the set suit entry;
        now the rank of item is the set rank entry.

Report forcing a hand:
    try taking inventory.

Example **The Trouble with Printing**

Making a READ command, distinct from EXAMINE, for legible objects.

"The Trouble with Printing"

A thing has some text called printing. The printing of a thing is usually "blank".

Understand the command "read" as something new. Understand "read [something]" as reading. Reading is an action applying to one thing, requiring light. Check reading: if the printing of the noun is "blank", say "Nothing is written on [the noun]." instead. Carry out reading: say "You read: [printing of the noun][line break]". Report reading: do nothing.

The Archive is a room.

Berkeley's report is a thing in the Archive. The description is "A report from Governor Sir William Berkeley of Virginia, in 1671, in answer to the queries sent by the Commissioners of Plantations the year previous. Of this report the better part is burned and only a tail fragment remains." The printing of Berkeley's report is "I thank God, [italic type]there are no free schools[roman type] nor [italic type]printing[roman type], and I hope we shall not have these hundred years; for [italic type]learning[roman type] has brought disobedience, and heresy, and sects into the world, and [italic type]printing[roman type] has divulged them..."

Test me with "examine report / read report".

Since we defined reading as an action requiring light, we could further distinguish reading and examining (if we wanted) by writing some different visibility rules for it.

A book with pages that can be read by number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

Suppose we have a book that the player must consult page-by-page, and we want to be able to accept all of the following input:

```
> READ BOOK (to choose a random page and read it)
> READ PAGE 1 IN BOOK
> READ PAGE 2
> READ THE LAST PAGE OF THE BOOK
> READ THE NEXT PAGE
> READ PREVIOUS PAGE IN BOOK
> READ THE FIRST PAGE
```

One approach would be to write many different understand rules and actions: one action for reading randomly, one for reading a specific page, one for reading the first page, one for reading the previous page, one for reading the next page, and one for reading the last page. But this gets tedious to construct and maintain.

More usefully, we could consider that all of the last four options are essentially the same action at heart: the player is asking to read a page in the book using a name rather than a number, and we will have to perform a minor calculation to discover what the number should be. Here's an implementation using named values to resolve this problem:

```
"Pages"

The Library is a room. The sinister book is carried by the
player. The sinister book has a number called the last
page read. The sinister book has a number called the
length. The length of the sinister book is 50.

Understand the command "read" as something new.
```

Understand "read [something]" or "consult [something]" or "read in/from [something]" as reading. Reading is an action applying to one thing, requiring light.

Understand "read [number] in/from/of [something]" or "read page [number] in/from/of [something]" or "look up page [number] in/from/of [something]" or "consult page [number] in/from/of [something]" as reading it in. Reading it in is an action applying to one number and one thing, requiring light.

Named page is a kind of value. The named pages are first page, last page, next page, previous page.

To decide what number is the effective value of (L - last page):
    decide on the length of the book.

To decide what number is the effective value of (F - first page):
    decide on 1.

To decide what number is the effective value of (N - next page):
    let X be the last page read of the book plus 1;
    decide on X.

To decide what number is the effective value of (P - previous page):
    let X be the last page read of the book minus 1;
    decide on X.

Understand "read [named page] in/from/of [something]" or "read the [named page] in/from/of [something]" as

reading it relatively in. Reading it relatively in is an action applying to one named page and one thing, requiring light.

Does the player mean reading something in the sinister book: it is very likely.

This is the book requirement rule:
    if the player is not carrying the sinister book, say "You're not reading anything." instead.

Check reading it relatively in:
    if the second noun is not the sinister book, say "There are no pages in [the second noun]." instead;
    abide by the book requirement rule.

Carry out reading it relatively in:
    let N be the effective value of the named page understood;
    now the number understood is N;
    try reading N in the book.

Check reading it in:
    if the second noun is not the sinister book, say "There are no pages in [the second noun]." instead;
    abide by the book requirement rule.

Check reading it in:
    if the number understood is greater than the length of the sinister book, say "There are only [length of sinister book in words] pages in the book." instead;
    if the number understood is less than 1, say "The page numbering begins with 1." instead.

Carry out reading it in:
    read page number understood.

Check reading:
    if the noun is not the sinister book, say "There are no pages in [the noun]." instead;
    abide by the book requirement rule.

Carry out reading:
    let N be a random number between 1 and the length of the sinister book; now the number understood is N;
    say "You flip the pages randomly and arrive at page [the number understood]:[paragraph break]";
    try reading the number understood in the sinister book.

Table of Book Contents

page content
2 "dhuma jyotih salila marutam / samnipatah kva meghah / samdesarthah kva patukaranaih / pranibhih prapaniyah"
13 "lamathesteron pws eipe kai saphesteron"
50 "rgshilz maplot..."

To read page (N - a number):
    now the last page read of the sinister book is N;
    if there is a content corresponding to a page of N in the Table of Book Contents:
        choose row with a page of N in the Table of Book Contents;
        say "You read: '[content entry]'[paragraph break]";
    otherwise:
        say "Page [N] appears to be blank."

To read page (N - 47):
    say "Your eyes burn; your ears ring. Beneath your gaze, the dreadful sigils writhe, reminding you of that which lies outside the edges of the universe...";

```
    end the story saying "You have lost your remaining
sanity".

Test me with "read from the sinister book / read the
book / read the next page / read page 2 / read previous
page / g / read the first page / read the last page of the
book / read the next page / read 47 in book".
```

---

443 **Example** **The Fourth Body** *WI*

Notebooks in which the player can record assorted notes throughout play.

Some mystery games supply the player with an in-game system for taking notes, in case he doesn't want to rely on scraps of paper next to the computer. One way of doing this is to write out all the player's notes and observations into a notebook file, whose contents can be retrieved during play (or, indeed, after it).

We'll first invent a general system for writing text into notebooks, by creating a new kind called jotter. Each individual jotter will have its own disc file, and there will be basically three things which can be done with jotters: erasing, reading and writing.

```
"The Fourth Body"

A jotter is a kind of thing. A jotter has an external file
called the text file. A jotter can be fresh or used. A jotter
is usually fresh. A jotter has a text called the heading.

The currently erased jotter is an object that varies.
To erase (pad - a jotter):
    now the currently erased jotter is the pad;
    write "[heading of the currently erased jotter]
[paragraph break]" to the text file of the pad;
    now the pad is fresh.
```

```
To write in (pad - a jotter):
    append "[the time of day]: [topic understood][line
break]" to the text file of the pad;
    now the pad is used.


To read (pad - a jotter):
    say "You read: [paragraph break][text of the text file
of the pad]".
```

This is all as might be expected, except perhaps for the business of the "currently erased jotter". Why copy "pad" into this - why not simply write "[heading of the pad]"? The answer is that "pad" is a temporary "let" value, and cannot be used inside other phrases, such as the "write ... to ..." phrase.

We want to erase any jotters when play begins, as otherwise text left over from any previous games will still be visible:

```
When play begins:
    repeat with pad running through jotters:
        erase the pad.
```

Now we need to create rules to allow the player to control reading, writing and erasing. Reading we will handle with the ordinary examining action, but we create new actions for writing and erasing. A nice little trick allows WRITE WHATEVER to default to writing WHATEVER in a notebook being carried.

```
Instead of examining a used jotter (called the pad):
    read the pad.


Instead of examining a fresh jotter (called the pad):
    say "There is nothing of note in [the pad]."


Understand "write [text] in [something preferably held]"
as writing it in. Understand "write [text]" as writing it in.
```

```
Writing it in is an action applying to a topic and one
thing. Rule for supplying a missing second noun while
writing: if a jotter (called the pad) is carried, now the
second noun is the pad; otherwise say "You will have to
specify what to write that it."


Check writing it in:
    if the second noun is not a jotter, say "It would be
better to write in a notebook." instead.


Carry out writing it in:
    write in the second noun.


Report writing it in:
    say "Under the current time, you write '[the topic
understood]' into [the second noun]."


Understand "erase [something preferably held]" as
erasing. Erasing is an action applying to one carried
thing.


Check erasing:
    if the noun is not a jotter, say "It's hard to see how."
instead.


Carry out erasing:
    erase the noun.


Report erasing:
    say "You scrub out all the entries in [the noun]."
```

That completes a general-purpose implementation of jotters, and we put it to use:

The player carries a jotter called your notebook. The file of Player's Observations is called "notebook". The text file of your notebook is the file of Player's Observations. The heading of your notebook is "Observations in the Pottingham Green Case".

The Damp Hillside is a room. "It is just after dawn: among the trees there is misty and pale blue light. [if Havers is in the location]The only saturated color in view is the orange-and-yellow jacket of [Detective Havers]. She is trying unsuccessfully to light a cigarette. [end if] [paragraph break]The body itself is further down, closer to the bottom of the ravine. It would be foolish to speculate before seeing it, but the odds are that the corpse was rolled down after death. The ground is not steep enough for the fall itself to be deadly."

Detective Havers is a woman in the Damp Hillside. The description is "She gives you a weak smile when you look at her: you know she hasn't slept more than three hours any of the last few nights." Havers is scenery.

Havers is carrying a jotter called Barbara's notebook. The file of Barbara's Observations is called "barbara". The text file of Barbara's notebook is the file of Barbara's Observations. The heading of Barbara's notebook is "I could murder a cup of tea".

The time of day is 6:32 AM.

Instead of examining your notebook when your notebook is fresh:
    say "Your notebook is blank. Back in the office, of course, there are a stack of others. But you brought a fresh notebook in a kind of weary hope. You're going to pretend, just for now, that this body might be unrelated to the graphic string of murders you're already

investigating."

 Example Bibliophilia

A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

Suppose we want a bookshelf with a very large number of books on it. They aren't to be taken or carried around in the game, but they should be mentioned, and the player should be allowed to look them up by name. Furthermore, the player's attempts to examine something unrecognized should be understood as an attempt to look up a title -- but only when the player is in the presence of the books. The rest of the time such requests should be rejected in the usual way.

"Bibliophilia"

The Graduate Lounge is a room. "Shabby sofas; plastic cups remaining from the afternoon's pre-lecture espresso; a collection of Xena and Hercules figurines posed for ironic effect. It's somewhat depressing at this hour, when everyone has gone home."

The Classics Reading Room is south of the Lounge. "Not as large a collection as the one in the Library, but it contains copies of everything really essential for reference."

Understand "examine [text]" as examining as a book when the player is in the Reading Room. Understand "look up [text]" as examining as a book when the player is in the Reading Room.

Examining as a book is an action applying to one topic.

```
Carry out examining as a book:
    say "You can't find any such text."


Instead of examining as a book a topic listed in the
Table of Book Titles:
    say "[description entry][paragraph break]"


Table of Book Titles

title                description
"Reading Greek"      "A beginner's reading guide to Greek."
"Death and Dying"    "Reading about Death and Dying in preparation for reading Greek eschatology."
"TAPA Titles"        "A list of titles from the American Philological Association from 2004."
"Oxford Classical Dictionary"  "A 1949 dictionary of classical Greek and Roman culture, from meter to ancient cosmetics."
"Collected Dialogues of Plato"  "Edited by Edith Hamilton and Huntington Cairns" translations -- but still a useful single volume, ed. Edith Hamilton and Huntington Cairns."
"Adobe Illustrator CS User Guide"  "A useful guide for Adobe Illustrator." illustrator"  "adobe/illustrator/user/guide"


Some books are scenery in the Reading Room.
Understand "copies" or "book" or "shelf" or "shelves" as
the books. Instead of examining the books:
    choose a random row in the Table of Book Titles;
    say "You scan the shelves and notice, among others,
a volume entitled [italic type][title entry][roman type]."


Test me with "south / examine ocd / examine books /
examine books / examine plato / n / x hercules / s / x
hercules".
```

Now if we type >X HERCULES in the Lounge, we will get

```
>x hercules
```

```
You can't see any such thing.
```

thanks to our somewhat slovenly implementation of the Lounge scenery; but in the Reading Room,

```
>x hercules
You can't find any such text.
```

In practice we might also want to extend our coverage somewhat to handle a case where the player tried to take books from the bookshelf: currently that would not be understood.

---

335 **ExampleAARP-Gnosis** *WI*

An Encyclopedia set which treats volumes in the same place as a single object, but can also be split up.

Suppose we have a complete Encyclopedia in our game. The player is allowed to pick up the whole set (there must not be too many volumes), but also to do things with individual volumes, and indeed to scatter these volumes all over the place. Putting a volume back in the same place as the rest of the Encyclopedia should, however, restore it to the collective. We will start out by defining general rules for collectives like this:

```
"AARP-Gnosis"
```

```
Fitting relates various things to one thing (called the
home). The verb to fit means the fitting relation.
Definition: a thing is missing if it is not part of the home
of it.
```

```
A collective is a kind of thing.
```

```
Before doing something to something which is part of a
collective:
```

```
    let space be the holder of the home of the noun;
    move the noun to the space.


Instead of examining a collective:
    say "[The noun] consists of [the list of things which
are part of the noun]."
```

Now the real work begins. One reason to make this an activity is that we might easily want to override it for specific objects; for instance, the generic collecting activity here would not deal properly with collectives of clothing where some items might be worn and others not. In that case, we would want to write another, more specific "collecting" activity to handle the complexities of fashion.

```
Collecting something is an activity.


Every turn:
    repeat with item running through collectives:
        carry out the collecting activity with the item.


To remove (item - a thing) when empty:
    let space be the holder of the item;
    if the number of things which are part of the item is 0:
        now the item is nowhere;
    if the number of things which are part of the item is 1:
        let the last thing be a random thing which is part of
the item;
        move the last thing to the space;
        now the item is nowhere.


Before collecting a thing (called the item):
    remove item when empty;
    let space be the holder of the item;
    if space is not a thing and space is not a room:
        if something (called the other space) contains at
least two things which fit the item, move item to the
other space;
        if a room (called the other space) contains at least
two things which fit the item, move item to the other
```

```
space;
        if someone (called the owner) carries at least two
things which fit the item, move item to the owner.
```

```
Rule for collecting a thing (called the item):
    let space be the holder of the item;
    if space is a thing or space is a room:
        repeat with component running through things held
by the space:
            if the component fits the item, now the
component is part of the item;
        remove item when empty.
```

And now for a cheerful scenario:

```
The Boise Memorial Library is a room. "A concrete box
of a room, roughly eight feet by fourteen, which contains
all the fallout shelter has to offer by way of
entertainment. Someone with a grim sense of humor
has tacked a READ! literacy poster to the door, as
though there were anything else to do while you await
the calming of the Geiger counters." The shelf is a
supporter in the Library. "A battered utility shelf stands
against the south wall."
```

```
The New Idahoan Encyclopedia Set is a collective.
Volume A-Aalto fits the Encyclopedia. It is part of the
Set. Volume AAM-Aardvark fits the Encyclopedia. It is
part of the Set. Volume Aarhus-Aaron fits the
Encyclopedia. It is part of the Set. Volume AARP-
Gnosis fits the Encyclopedia. It is part of the Set.
Volume Gnu-Zygote fits the Encyclopedia. It is part of
the Set. The Set is on the shelf.
```

Let's have the Encyclopedia describe itself differently depending on whether it's all in one place:

```
After printing the name of the Set when something
missing fits the Set:
    say " (missing [a list of missing things which fit the
Set])"
```

```
Before printing the name of the Set when the number of
missing things which fit the set is 0:
    say "complete ".
```

```
Test me with "get aarhus-aaron / look / inventory / get
aam-aardvark / look / get gnu-zygote / look / get aarp-
gnosis / look / inventory / drop set / look / get set / get a-
aalto / inventory".
```

Example **The Fifth Body**

An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

The implementation here is much like that of the previous example, except that we allow the player to write his notebook input as a separate command, leading to an exchange such as

```
>write in my notebook
You open your notebook and prepare to write in it.
```

```
>>Am beginning to think that HT and BGG are in this
together.
You finish writing and fold your notebook away.
```

```
>read my notebook
You read:
```

```
Wednesday morning
```

Am beginning to think that HT and BGG are in this together.

The opening is much as before:

"The Fifth Body"

A jotter is a kind of thing. A jotter has an external file called the text file. A jotter can be fresh or used. A jotter is usually fresh. A jotter has a text called the heading.

The currently erased jotter is an object that varies.

To erase (pad - a jotter):
    now the currently erased jotter is the pad;
    write "[heading of the currently erased jotter][paragraph break]" to the text file of the pad;
    now the pad is fresh.

To write in (pad - a jotter):
    append "[the time of day]: [player's command][line break]" to the text file of the pad;
    now the pad is used.

To read (pad - a jotter):
    say "You read: [paragraph break][text of the text file of the pad]".

When play begins:
    repeat with pad running through jotters:
        erase the pad.

```
Instead of examining a used jotter (called the pad):
    read the pad.

Instead of examining a fresh jotter (called the pad):
    say "There is nothing of note in [the pad]."

Target jotter is an object that varies. The target jotter is
usually nothing.

Understand "write in [something preferably held]" as
writing in. Writing in is an action applying to one thing.

Check writing in:
    if the noun is not a jotter, say "It would be better to
write in a notebook." instead.

Carry out writing in:
    now the command prompt is ">>";
    now the target jotter is the noun.

Report writing in:
    say "You open [the noun] and prepare to write in it."
```

Now what happens is that the player, having typed WRITE IN NOTEBOOK, will be faced with a ">>" prompt instead of the usual ">": a sign that the input mode has changed.

The next code is to react to reading a command. Whatever the player types at the >> prompt when the target jotter is set will now be recorded in the notebook, though with a character limit of about 60-100 characters depending on how much upper-case and punctuation he uses. (There are ways to lift the character length restriction as well, but they would take us into deeper waters.)

After reading a command when target jotter is a jotter:
    now the command prompt is ">";
    write in target jotter;
    now target jotter is used;
    say "You finish writing and fold your notebook away.";
    now the target jotter is nothing;
    reject the player's command.

Understand "erase [something preferably held]" as erasing. Erasing is an action applying to one carried thing.

Check erasing:
    if the noun is not a jotter, say "It's hard to see how." instead.

Carry out erasing:
    erase the noun.

Report erasing:
    say "You scrub out all the entries in [the noun]."

The player carries a jotter called your notebook. The file of Player's Observations is called "notebook". The text file of your notebook is the file of Player's Observations. The heading of your notebook is "Sunday Morning".

The Vestry is a room. "[Havers] hangs back by the door: the forensics expert is not finished with a preliminary examination of the body. From here you can't see much, except that the expert has peeled back and laid to one side a liturgical vestment that someone at the church used to cover the corpse until the police came. What was once a cream silk with festive Easter embroidery is now stained with blood-colored handprints."

Detective Havers is a woman in the Vestry. The description is "She looks glumly back. There's still a purple-ish bruise on her cheekbone from the disaster Thursday afternoon." Havers is scenery.

Havers is carrying a jotter called Barbara's notebook. The file of Barbara's Observations is called "barbara". The text file of Barbara's notebook is the file of Barbara's Observations. The heading of Barbara's notebook is "Sun. AM".

The time of day is 9:11 AM.

ExampleCosta Rican Ornithology

A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

The following relies on quite a number of features we haven't met yet: tables, rules for printing names, instructions for understanding the player's commands. It is offered simply as an example of how a fully implemented book might be handled in Inform.

"Costa Rican Ornithology"

A book is a kind of thing. Understand "book" as a book.
A book has a table name called the contents.

Instead of consulting a book about a topic listed in the contents of the noun:
    say "[reply entry][paragraph break]".

```
Report consulting a book about:
    say "You flip through [the noun], but find no reference
to [the topic understood]." instead.
```

With this "topic understood" phrase, we're telling Inform to print back the word or phrase that the player was attempting to look up. This overrides the more general default response, "You discover nothing of interest in the book."

We now have the essential elements to construct whatever books we like. Now let's have an example of a specific book:

```
The Guide to Central American Birds is a book carried
by the player. The contents of the Guide is the Table of
Listed Birds.
```

We will come back to the idea of tables and table names later, but for now the important thing is that we have instructed Inform to look up its answers to consulting the bird guide in this form:

```
Table of Listed Birds

topic                reply
"[red]"              "You flip through the Guide for a while and eventually discover a
                     reference to the [scarlet macaw], which appears to correspond
                     with what you see before you."
"quetzal"            "The quetzal is quite extraordinary, describing its brilliant
                     plumage, flashing and igniting in the sunshine, which is
                     supposedly sufficient to lure birdwatchers from all over the world.
                     Unfortunately, the quetzal is described as being bright emerald in
                     color, with a pink fuzz on its head and a long soft tail 'like a feather
                     boa'. None of these describes your visitor."
```

The topic column is a bit special: it matches the player's input, and is not meant to be printed out again. Topic columns will be discussed further in the chapter on Tables. (Note also that, however it may appear in the documentation, the topic column should not be spanning multiple lines in our source text.)

We may also compress long or complicated topics by creating bracketed abbreviations, and in fact it's useful to do so now, to explain the red token we just used:

```
Understand "red-orange" or "bird" or "red" or "orange"
as the scarlet macaw. Understand "red-orange" or "red"
or "orange" or "scarlet" as "[red]".
```

This technique is discussed further in the chapter on Understanding.

If we wanted more books, we could define those in the same way, giving each its own separate contents table to be used for consultation. But for the sake of the example we will keep it simple, and move on to the scenario itself:

```
The Veranda is a room. "From here you can see a
considerable expanse of dense-growing jungle plants,
and eventually the open water beyond."


The scarlet macaw is an animal in the veranda. "A
vibrantly-colored [scarlet macaw] perches on the rail."


A thing can be known or unknown.


Before printing the name of the scarlet macaw while
consulting:
    now the scarlet macaw is known.


Rule for printing the name of the unknown scarlet
macaw: if the macaw is unknown, say "red-orange bird
of unknown species".


Test me with "look up penguins in the guide / look up
quetzal in guide / look up silver nuthatches in the guide /
look / look up red bird in the book / look".
```

An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean") are accepted alongside everyday ones ("blue").

There are hundreds of traditional pigments, from lampblack to burnt sienna, so we will confine ourselves to just two:

"Palette"

The Atelier is a room. "The floridly untidy loft space used by a moderately unsuccessful artist (you, that is)." The canvas, palette and paint brush are here. Understand "painting" as the canvas.

Colour is a kind of value. The colours are white, red, blue and green.

The canvas has a colour. The canvas is white. The printed name of the canvas is "largely [colour] canvas".

Painting is an action applying to one thing and one colour. Check painting: if the noun is not the canvas, say "Centuries of tradition suggest that canvas is the natural home of paint." instead. Carry out painting: now the colour of the canvas is the colour understood. Report painting: say "You splash away at the now [canvas]."

Understand "paint [something] [a colour]" as painting.

Understand "calico" as white. Understand "cerulean" or

"cerulean blue" as blue.

Test me with "examine canvas / paint canvas red / examine canvas / paint canvas cerulean / examine canvas".

---

36 ExampleBrown *WI*

A red sticky label which can be attached to anything in the game, or removed again.

"Brown"

The Shipping Room is a room. The red sticky label is a thing carried by the player. The description of the red sticky label is "It reads: AIRMAIL[if the label is part of something (called the parent)]. It is stuck to [the parent][end if]."

A black crate is in Shipping. The description is "A boring black crate." The brown crate is a thing in Shipping. The description is "An ordinary brown crate."

After examining something when the label is part of the noun:
    say "A bright red sticky label is attached to [the noun]!"

Here is the essential point: whenever we ATTACH LABEL TO something, it becomes part of that object.

Instead of tying the red sticky label to something:
    now the red sticky label is part of the second noun;

```
        say "You stick [the label] to [the second noun]."
```

And of course the label cannot be stuck to itself or to more than one thing at a time.

```
Before tying the label to something when the label is
part of something:
    if the label is part of the second noun:
        say "[The label] is already stuck to [the second
noun]." instead;
    otherwise:
        say "(first freeing the label)[line break]";
        silently try taking the label;
        if the label is part of something, stop the action.


Instead of tying the red sticky label to the label:
    say "That would ruin the label entirely."


Instead of taking the label when the label is part of
something:
    now the player carries the label;
    say "You peel the label off again."
```

Much of the rest is just tidying to make sure that the player's commands are redirected into the right syntax.

```
Instead of tying something to the label:
    try tying the label to the noun.


Instead of putting the label on something:
    try tying the label to the second noun.


Instead of inserting the label into something:
    try tying the label to the second noun.
```

```
Understand the commands "stick" or "apply" as "tie".
```

We could have created a new "sticking" action, but to keep the example short we will use the built-in "tying" action instead, and respond to the command "stick" just as if it were "tie".

```
Understand "peel [something]" or "peel off [something]"
as taking.


Test me with "i / put label on the black crate / look / x
black / x label / get the label / apply label to brown crate
/ look / x brown / peel off label / stick label to label".
```

Example **Early Childhood**

55 *WI*

A child's set of building blocks, which come in three different colours - red, green and blue - but which can be repainted during play.

This would be a one-star example if it were not for the repainting:

```
"Early Childhood 1"


A building block is a kind of thing. A red block, a blue
block and a green block are kinds of building block.


The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.


Test me with "look / get red block".
```

But a kind cannot change during play, so this will not do. Instead, the colour will have to be a property of the block. So we might first try this:

```
"Early Childhood 2"

Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue.

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.

Test me with "look / get red block".
```

Which is fine, so far as it goes, but the colour property is not at all visible to the player, who simply sees "eleven blocks". We thought of colour as being something outwardly apparent, but Inform does not know this. To achieve a better effect, we will need features from distant chapters. The first is an activity called "printing the name of":

```
"Early Childhood 3"

Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue. Before printing the name of a
block: say "[colour] ". Before printing the plural name of
a block: say "[colour] ".

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.

Test me with "look / get red block".
```

This too, however, is unsatisfactory. The individual blocks are correctly

described, but we are unable to distinguish them during play: we cannot type "take a green block", for instance. And because the blocks are indistinguishable in play, they are still massed together as "eleven blocks" in room descriptions. We need to go one step further:

```
"Early Childhood 4"

Colour is a kind of value. The colours are red, blue and
green. A block is a kind of thing. A block has a colour. A
block is usually blue. Before printing the name of a
block: say "[colour] ". Before printing the plural name of
a block: say "[colour] ". Understand the colour property
as describing a block.

The Nursery is a room. In the Nursery are six red
blocks, four blue blocks and a green block.
```

And now everything works nicely: the blocks are grouped by colour, and can be referred to by colour, and we can even change the colour of an individual block during play, using a bit of extra trickery from later:

```
Understand "paint [something] [colour]" as painting it.
Painting it is an action applying to one thing and one
colour. Check painting it: if the noun is not a block, say
"Paints are only for blocks." instead. Carry out painting
it: now the colour of the noun is the colour understood.
Report painting it: say "The block is now [the colour of
the noun]."

Test me with "get red block / get blue block / g / i / look /
paint blue block red / i / look / paint me red".
```


Objects which automatically include a description of their component parts whenever they are examined.

It is straightforward to make a rule that anything with parts must mention all those parts during an EXAMINE command:

"Control Center"

After examining a thing when something is part of the noun:
    say "[The noun] includes [a list of things which are part of the noun]."

The Control Center is a room. "Here you are at the Control Center of the universe."

The Universe Management Computer is a fixed in place thing in the Control Center. "The Universe Management Computer sits directly before you, unguarded." The description of the Universe Management Computer is "The computer is so large that you would be unable to operate it all from one position. Alas, it does not come with a manual."

A chartreuse indicator light, an ennui meter, a golden knob settable to 15,000 positions, a toothpick dispenser, and a button labeled RESTART are part of the Universe Management Computer.

The command chair is an enterable supporter in the Control Center. It is pushable between rooms. "Because the computer is too large for you to reach all of the front panel from a standing position, there is a command chair on casters which allows you to push back and forth." The description of the command chair is "Quite ordinary, really, but for the heady rush of power that comes of sitting in it.". Some casters are part of the command chair.

Now whenever we look at any object with components, we will first see the description, then a list of parts which belong to it. The following refinement brings in elements of later chapters, but it may be worth noting: because we've written our rule as an "After examining...", anything that pre-empts the operation of the examine command will also prevent that rule from occurring. So for instance:

```
A hair-thick needle is part of the ennui meter.

Instead of examining the ennui meter: say "You can't be
bothered."
```

...would not result in the needle being mentioned.

```
Test me with "x chair / x computer / x ennui meter"
```

Example **What Makes You Tick** *WI*

Building a fishing pole from several component parts that the player might put together in any order.

Suppose we want to let the player build a fishing pole out of three parts: a hook, a string, and a stick.

There are several things we must account for here. One is that our combination verb should be insensitive to ordering: it shouldn't matter whether the player types COMBINE STICK WITH STRING or COMBINE STRING WITH STICK.

Second, we need to make sure that our implementation handles intervening stages of assembly gracefully. The player should be able to combine string and hook first, or string and stick first, and be able to complete the assembly in either case.

Our implementation here uses a table of lists to determine which combinations of inputs should produce which result object. Because we sort our lists before comparing them, we guarantee that the player's ordering doesn't matter: COMBINE STICK WITH STRING will have the same effect as COMBINE STRING WITH STICK.

What's more, our implementation could be expanded to account for many other assemblages, if we wanted object-building to be a running theme of puzzles in our game.

```
"What Makes You Tick"


Understand "combine [something] with [something]" as
combining it with. Combining it with is an action applying
to two carried things. Understand the command
"connect" as "combine".


Understand the command "attach" as something new.
Understand "attach [something] to [something]" as
combining it with.


The combining it with action has an object called the
item built.


Setting action variables for combining something with
something:
    let X be a list of objects;
    add the noun to X;
    add the second noun to X;
    sort X;
    repeat through the Table of Outcome Objects:
        let Y be the component list entry;
        sort Y;
        if X is Y:
            now the item built is the result entry.


Check combining it with:
    if the item built is nothing or the item built is not in
limbo,
        say "You can't combine [the noun] and [the second
noun] into anything useful." instead.
```

```
Carry out combining it with:
    move the item built to the holder of the noun;
    now the noun is nowhere;
    now the second noun is nowhere.

Report combining it with:
    say "You now have [an item built]."

Limbo is a container. Limbo contains a hookless fishing
pole, a hooked line, and a complete fishing pole.

Streamside is a room. The player carries a stick, a wire
hook, and a string.

Table of Outcome Objects

result                component list
hookless fishing pole {stick, string}
hooked line           {wire hook, string}
hooked fishing pole   {hooked line, stick}
complete fishing pole {hookless fishing pole, wire hook}

Test me with "combine stick with string / i / combine
pole with hook / i".
```

This kind of implementation makes sense if we don't intend the player to take the fishing pole apart again, or to refer to any of its component parts once it is built. For an alternate approach that does allow assembled objects to be taken apart again, see "Some Assembly Required".

Suppose we're particularly mechanically-minded and would like a game in which all of our mechanical devices have buttons to turn them on and off.

```
"Model Shop"

An on/off button is a kind of thing.

Instead of pushing an on/off button which is part of a
switched off device (called the machine):
    try switching on the machine.
```

Here we are making a rule about how our hypothetical buttons will interact with the machines to which they belong. Instead of pushing... is a rule that pertains to actions, and we will learn more about these in the chapter on actions. "...which is part of a switched off device" provides a specific circumstance - this is only to apply to buttons that are stuck to a machines that can be turned on or off. "(called the machine)" tells Inform that if it finds such a device, it should thereafter refer to it as "the machine." (The called syntax is explained further in the chapter on Change.)

A set of three more rules will complete our instructions about using buttons to control devices:

```
Instead of pushing an on/off button which is part of a
switched on device (called the machine):
    try switching off the machine.

Instead of switching on an on/off button which is part of
a device (called the machine):
    try switching on the machine.
```

```
Instead of switching off an on/off button which is part of
a device (called the machine):
    try switching off the machine.
```

Then we hand out buttons with a free hand:

```
One on/off button is part of every device.
```

```
The Model Shop is a room. A model train is a fixed in
place device in the Model Shop. A toy elephant is a
device in the Model Shop.
```

```
Every turn when the model train is switched on:
    say "The model train circles your feet, blowing small
puffs of steam."
```

```
Every turn when the toy elephant is switched on:
    say "The toy elephant waves its trunk at you."
```

```
Test me with "push model train's button / push
elephant's button / g / switch off model train's button".
```

And now the game will have a model train's button and a toy elephant's button.

It may be that we want (as an added nuance) to add other names for these items. While we would want an assembly to create objects such as "Lucy's hand" and not "Lucy hand", it is entirely reasonable to want to talk about the model train button or the elephant button. We could define these additional names like so:

```
Understand "elephant button" or "button on elephant" as
the elephant's button.
```

```
Understand "model train" or "model" or "train" as
"[train]". Understand "[train] button" or "button on [train]"
as the model train's button.
```

In the second case, we are defining [train] to mean any of the three phrases "train", "model", and "model train"; so "[train] button" will match "model train button" or "train button" or "model button" equally well. See the chapter on Understanding for more on how to create alternative phrasings for the player to use.

Example **Signs and Portents** *WI*

53

Signpost that points to various destinations, depending on how the player has turned it.

```
"Signs and Portents"

Seven Dials is a room. The description of Seven Dials is
"There is a signpost, on which seven hands swivel and
swing, freely as weathercocks. They make your present
road now London, now Abingdon; now Weston-super-
Mare, or now Hell."

Seven Dials contains a signpost. The signpost is
scenery. Understand "sign" and "post" as the signpost.

Destination is a kind of value. The destinations are
London, Abingdon, Luton, Weston-super-Mare,
Runnymede, Hell, and Low Noon.

The signpost has a destination.
```

In order to interact with the signpost, we will need to make use of some

action rules:

Instead of examining the signpost:
    say "[The signpost] currently puts you on the road to [italic type][the destination of the signpost][roman type], but it swiftly alters again.";
    now the destination of the signpost is the destination after the destination of the signpost.

Instead of turning the signpost:
    now the destination of the signpost is the destination after the destination of the signpost;
    say "With a hand's touch you turn the signpost to mark your way for [italic type][the destination of the signpost][roman type]."

Instead of going north in Seven Dials when the destination of the signpost is Hell:
    say "It is a path that goes gently ever down and down with no stumbling block or any distraction at either side; there are no bandits and no tolls.";
    end the story.

Instead of going north in Seven Dials when the destination of the signpost is Low Noon:
    say "A long road whose scenery does not change, nor anything on the horizon move but the sun. When at last you come to Noon, she hangs above your head like a hat.";
    end the story finally.

Test me with "x signpost / n / turn signpost / n / turn signpost / n / turn signpost / n / turn signpost / n".

Test more with "x signpost / n / turn signpost / n / turn signpost / n / turn signpost / n / turn signpost / turn signpost / n".

Example **Radio Daze**

A radio that produces a cycle of output using varying text.

"Radio Daze" by Jon Ingold

The Living Room is a room. "A long couch, set up so you can see your wireless set. Not that you need to see it, of course."

The long couch is an enterable scenery supporter in the living room. Instead of entering the long couch when the radio was switched off, say "Better turn the radio on before you get comfortable." Report entering the couch: say "You settle yourself down to listen." instead. Instead of listening when the radio is switched on, stop the action.

The radio is a device in the living room. The radio is switched off. "[if switched on]The radio burbles on[otherwise]The radio is off[end if]." Check switching off the radio when the player is on the long couch: say "You can't reach the radio from here." instead.

Every turn when the radio is switched on:
    say "[one of]Two characters in the radio play have begun an argument[or]The argument continues[or]The play continues[stopping]: [one of]'Did not!'[or]'Did too!'[or]'Did I?'[or]'You did!'[or]'I couldn't have, Martha!'[or]'But you did, Tom!'[cycling]"

Test me with "sit on couch / turn on radio / sit on couch/ listen / g / g / g / g / g".

Understanding aspect ratios (a unit) in the names of televisions.

Named properties are not the only kind that Inform is able to understand referring to an object. We can also use unit and number properties to distinguish things from one another, as here, where televisions have aspect ratios:

```
"Aspect"

An aspect ratio is a kind of value. 16:9 specifies an aspect ratio.

A television is a kind of device. A television has an aspect ratio. Understand the aspect ratio property as referring to a television. Understand "European standard" as 16:9.

The Office is a room.

The widescreen TV is a television in the Office. The fifties TV is a television in the Office. The widescreen TV is 16:9. The fifties TV is 4:3.

Test me with "examine european standard tv / x 16:9 tv / x 4:3 tv".
```

We might want to allow every television to be tuned to a channel (a number property) which the player could refer to, so that

```
WATCH CHANNEL 13
TURN OFF CHANNEL 4
```

would be directed to the appropriate television object, if any television is turned on and tuned to the correct station. We might now write:

```
"Channel"

A television is a kind of device. A television has a
number called the channel. Understand the channel
property as referring to a television. Understand
"channel" as a television.

The Office is a room. The widescreen TV is a television
in the Office. The fifties TV is a television in the Office.

Changing the channel of it to is an action applying to
one thing and one number.

Understand "tune [something] to [number]" or "change
channel of [something] to [number]" as changing the
channel of it to.

Check changing the channel of something to:
    if the noun is not a television, say "[The noun] cannot
be tuned to a channel." instead.
```

```
Carry out changing the channel of something to:
    now the channel of the noun is the number
understood.



Report changing the channel of something to:
    say "You tune [the noun] to channel [number
understood]."



Instead of examining a television:
    if the noun is switched off, say "[The noun] is
currently turned off." instead;
    let the chosen channel be the channel of the noun;
    if the chosen channel is a current channel listed in the
Table of Television Channels:
        choose row with current channel of the chosen
channel in the Table of Television Channels;
        say "[output entry][paragraph break]";
    otherwise:
        say "Snow fills the screen of [the noun]."



Table of Television Channels

output channel
0 "The screen of [the noun] is completely black."
4 "A gloomy female news anchor describes the latest car bomb in
Baghdad: 104 dead today, and no sign of change."
5 "A couple of contestants in spangled scarlet outfits are performing
an energetic paso doble."
10 "On-screen, Ichiro is up to bat with one man on second and no
outs."



Test me with "change channel of fifties tv to 4 / x
channel 4 / switch on fifties / x channel 4 / switch on
widescreen / tune fifties tv to 5 / x channel 5 / x fifties tv /
x channel 4".
```

ExampleChannel 2

Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

Our previous implementation of televisions ("Channel 1") doesn't allow the player to type things like

TUNE FIFTIES TELEVISION TO CHANNEL 4

nor does it deal with player input like

TUNE TO CHANNEL 4 ON FIFTIES TELEVISION

or

TUNE TO CHANNEL 4

where no television is specified. When we are designing commands which involve two elements (here, a television and a channel number), it's usually a good idea to allow the player to specify those elements in either order, as we saw demonstrated briefly in "New commands for old grammar".

We might, therefore, want to add a few refinements: first by defining a "[channel]" token that will accept input of the forms "[number]" and "channel [number]", and second by creating some additional "Understand" lines that will accept variant versions of the player's input.

"Channel 2"

Section 1 - Televisions in General

A television is a kind of device.

A television has a number called the channel. Understand the channel property as referring to a television. Understand "channel" as a television.

Changing the channel of it to is an action applying to one thing and one number.

Understand "tune [television] to [channel]" or "change channel of [television] to [channel]" as changing the channel of it to.

Understand "tune [something] to [channel]" or "change channel of [something] to [channel]" as changing the channel of it to.

Understand "tune to [channel] on [television]" or "change to [channel] on [television]" as changing the channel of it to (with nouns reversed).

Understand "tune to [channel] on [something]" or "change to [channel] on [something]" as changing the channel of it to (with nouns reversed).

Understand "[number]" or "channel [number]" as "[channel]".

Check changing the channel of something to:
    if the noun is not a television, say "[The noun] cannot be tuned to a channel." instead.

```
Carry out changing the channel of something to:
    now the channel of the noun is the number
understood.



Report changing the channel of something to:
    say "You tune [the noun] to channel [number
understood]."



Instead of examining a television:
    if the noun is switched off, say "[The noun] is
currently turned off." instead;
    let the chosen channel be the channel of the noun;
    if the chosen channel is a current channel listed in the
Table of Television Channels:
        choose row with current channel of the chosen
channel in the Table of Television Channels;
        say "[output entry][paragraph break]";
    otherwise:
        say "Snow fills the screen of [the noun]."



Table of Television Channels

current channel    output
0                  "The screen of [the noun] is completely black."



Section 2 - The Scenario



The Office is a room.



The widescreen TV is a television in the Office. The
fifties TV is a television in the Office.
```

And we add the scenario-specific content to our Table of Television Channels; in the case of channel 13, we provide for a changing sequence of events using text variations.

Table of Television Channels (continued)

output
4 "A gloomy female news anchor describes the latest car bomb in Baghdad: 104 dead today, and no sign of change."
5 "A couple of contestants in spangled scarlet outfits are performing an energetic paso doble."
13 "[one of]On-screen, Ichiro is up to bat with one man on second and no outs.[or]Ichiro has singled to first and the other man is on third.[or]The next batter is in the middle of flying out.[or]Everything looks rosy until the men in black pull off a double-play and retire the side.[or]The channel has cut to a commercial.[stopping]"

Test me with "test one / test two".

Test one with "change channel of fifties tv to 4 / x channel 4 / switch on fifties / x channel 4 / switch on widescreen / tune fifties tv to channel 5 / x channel 5 / x fifties tv / x channel 4".

Test two with "tune to channel 13 / widescreen / tune channel 13 to channel 5 / tune channel 5 to channel 3 / widescreen / x channel 3".

Example Aftershock

Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times.

The built-in behavior of Inform is to print a line after a device is examined, saying whether the item is on or off. This is often inappropriate, and we could simply turn off that behavior in general by instructing Inform to ignore the "examine described devices rule" (see the chapter on rulebooks).

Perhaps, though, we would like continue to have a short passage about

the action of any switched on device; we'd just like a little more control over what it says from time to time. And in that case, we might change the rule to give a new activity control over that portion of the description:

```
"Aftershock"

Section 1 - Showing actions

Showing action of something is an activity.

Rule for showing action of something (called item):
    if the item is switched on, say "[The item] is switched
on.";
    otherwise say "[The item] is switched off."
```

Borrowing from the rulebooks chapter, we can replace the standard "examine described devices" rule with something that uses this activity.

```
The new described devices rule is listed instead of the
examine devices rule in the carry out examining rules.

This is the new described devices rule:
    if the noun is a device:
        carry out the showing action activity with the noun;
        now examine text printed is true.
```

Thus far we have essentially replicated the original behavior, but we've made it possible to write specialized behavior for devices, and to invoke that behavior in other places:

```
Report switching on something:
    say "You flip a switch. ";
    carry out the showing action activity with the noun
instead.
```

This might be useful for an electric lamp kind:

Section 2 - Electric Lamps

An electric lamp is a kind of device.

Rule for showing action of an electric lamp (called item):
    if the item is switched on, say "[The item] is lit[if the number of visible lit things is greater than 1], competing with [the list of visible lit things which are not the item] [end if].";
    otherwise say "[The item] is dark."

Carry out switching on an electric lamp: now the noun is lit. Carry out switching off an electric lamp: now the noun is unlit.

Section 2 - The Scenario

The time of day is 3:47 AM. When play begins, now the right hand status line is "[time of day]".

The Downstairs Hallway is a dark room. "The only room in the house with no furniture and almost nothing on the walls. At times like this you always notice the crack in the plaster, originating near the light fixture and running almost all the way to the wall."

A plastic jug of filtered water is in the Downstairs Hallway. The description is "Five gallons, not that that will last you very long, hot as it has been lately."

```
The crack is scenery in the Hallway. The description is
"No, the ceiling isn't going to fall on you today."
```

```
The light fixture is an electric lamp in the Hallway. It is
switched on, lit, and scenery. The description is "A plain
globe of frosted glass containing the light bulb. Nothing
special, and you never think about it except when, as
now, you are forced to spend hours in this room."
```

```
The flashlight is an electric lamp carried by the player.
The description is "A shiny red flashlight." The portable
radio is a device carried by the player. The description is
"A small battery-operated radio which you received for
free with your subscription to US News & World Report.
It has served you well through many earthquakes past."
```

And with our activity, we can override the flashlight's electric lamp behavior with new behavior:

```
Rule for showing action of the flashlight:
    if the flashlight is switched on, say "A strong, narrow
beam of light shines from the flashlight.";
    otherwise say "It is currently switched off."
```

...or give special actions for the radio:

```
Rule for showing action of the radio:
    if the radio is switched on, say "Through the static,
you pick up pieces of discussion: a 6.7 on the Richter
scale, epicenter... something about Topanga... but it
crackles out again.";
    otherwise say "The radio is silent. You're saving the
batteries."
```

```
Instead of listening in the presence of the switched on
radio:
    carry out the showing action activity with the radio
```

instead.

Test me with "examine light / switch light off / switch flashlight on / switch radio on / examine radio / examine flashlight".

Example **Four Cheeses**

WI

A system of telephones on which the player can call distant persons and have conversations.

"Four Cheeses"

Section 1 - Telephones and Connections

A telephone is a kind of thing. Understand "phone" as a telephone.

Understand "call [any telephone] on [something]" as calling it on. Understand "call [any telephone]" as calling it on. Understand the commands "dial" or "phone" or "telephone" as "call". Understand "call [any known person]" as calling it by name on.

Connection relates one thing to another (called the other party).

The verb to reach means the connection relation.

```
Calling it on is an action applying to one visible thing
and one thing.


Check calling it on:
    if the second noun is not a telephone, say "[The
second noun] is unlikely to be much use in that respect."
instead;
    if the second noun is the noun, say "You get a busy
signal." instead.


Carry out calling it on:
    if a person (called the listener) can see the noun, now
the player reaches the listener.
```

Because we've said that connection is a reciprocal, one-to-one relationship, Inform will do the rest of the bookkeeping: if (for instance) we telephone someone else, the first connection will be broken automatically.

```
Report calling it on:
    say "'Hello?' says [the other party of the player]."
```

To avoid annoyance, we should also let the player use CALL #### as well as CALL #### ON TELEPHONE. A rule from the chapter on Activities comes in handy here:

```
Rule for supplying a missing second noun while calling
something on:
    assign a phone.


To assign a phone:
    if the player can touch a telephone (called the current
phone):
        say "(on [the current phone])[line break]";
        now the second noun is the current phone;
    otherwise:
        say "You don't have a phone handy."
```

Things might be a little more complicated if we had cell phones that could

be moved around, but for right now the player can only touch a maximum of one phone at a time.

Suppose we further want to allow the player to call people up by name, but only if they've already been encountered or are familiar to the player for some reason.

```
A person can be known or unknown.

Understand "call [any known person]" as calling it by
name on.

Understand "call [any known person] on [something]" as
calling it by name on.

Rule for supplying a missing second noun while calling
something by name on:
    assign a phone.

Calling it by name on is an action applying to one visible
thing and one thing.

Check calling it by name on:
    if the noun is in the location, say "[The noun] is right
here." instead.

Carry out calling it by name on:
    if the noun can touch a telephone (called the link), try
calling the link on the second noun;
    otherwise say "You can't reach [the noun]." instead.

Before calling something on something when the player
```

```
reaches someone:
    say "(first ending your conversation with [the other
party of the player])[command clarification break]";
    end current conversation.


Understand "hang up [something]" as hanging up.


Hanging up is an action applying to one thing.


Check hanging up:
    if the noun is not a telephone, say "You can't hang up
[the noun]." instead;
    if the player does not reach someone, say "You're not
on the line with anyone." instead.


Carry out hanging up:
    now the player does not reach anyone.


Report hanging up:
    say "You put down [the noun], cutting the
connection."


Before going somewhere when the player reaches
someone:
    say "(first hanging up on [the other party of the
player])[command clarification break]";
    end current conversation.
```

And finally we want to make sure that calling random other numbers produces a sensible result:

```
Understand "call [text]" as misdialling. Misdialling is an
action applying to one topic. Carry out misdialling: say
"The phone rings and rings but no one answers."
```

```
Understand "call 911" or "call 999" or "call police" or
"call fire department" as a mistake ("After strict
warnings, you've given up making prank calls to
emergency services.").


Before misdialling when the player reaches someone:
    say "(first ending your conversation with [the other
party of the player])[command clarification break]";
    end current conversation.


To end current conversation:
    let the current phone be a random telephone which
can be touched by the player;
    silently try hanging up the current phone.


After deciding the scope of the player while the player
reaches someone:
    place the other party of the player in scope, but not its
contents.
```

A note about this scope addition: the player can refer to the other party whenever he has the other person on the phone. He can't, however, see or refer to anything that person might be holding or wearing, thanks to the "but not its contents" option.

Furthermore, the player can't actually do anything to that person that requires touching. That's because of the reaching inside rules, which govern whether the player can reach through intervening barriers such as rooms. (See the Advanced Actions chapter for more about changing reachability.) There are two things we might want to be careful about, though.

First, we should specifically disallow the player from looking at the person on the other end of the line. Since sight doesn't require touching, the reaching inside rules will not be consulted about a command such as EXAMINE BOSS or LOOK UNDER BOSS. We can, however, intervene in such cases using the visibility rules, which are consulted for any actions that "require light" (including EXAMINE and LOOK UNDER). Here again we borrow some options from the Advanced Actions chapter:

```
To decide whether acting through the line:
    if the noun is something and the location of the noun
is not the location of the player:
        yes;
    if the second noun is something and the location of
the second noun is not the location of the player:
        yes;
    no.


Visibility rule when acting through the line:
    there is insufficient light.


Rule for printing a refusal to act in the dark when acting
through the line:
    say "You're not on a video phone, so you can only
hear." instead.
```

Second, though the existing reaching inside rules are adequate to stop us from touching the person on the other end of the line, the response that's currently printed is a bit generic: it just says "You can't reach into [the room containing the person]." Let's add our own custom reply, instead:

```
A rule for reaching inside a room (called destination):
    if the other party of the player is enclosed by the
destination:
        say "Though you're on the line with [the other party
of the player], you can't physically reach to [the
destination].";
        deny access.
```

Section 2 - Conversation over the Phone, In General

This portion supplies a simple method of conversation; but we could substitute some completely different conversation system if appropriate. The effect of the telephones is that we are allowed to talk to characters in distant locations under certain circumstances, after which the usual conversation rules apply.

Instead of listening to a telephone when the player reaches someone:
    say "You can hear [the other party of the player] breathing."

Before listening to someone when the player cannot touch the noun:
    say "[The noun] is waiting for you to carry on the conversation." instead.

A person has a table name called chatter.

Before telling someone about something:
    try asking the noun about it instead.

Before answering someone that something:
    say "Best to confine your conversation to questions and answers." instead.

Before asking someone about something:
    if the topic understood is a topic listed in the chatter of the noun, say "[reply entry][paragraph break]" instead;
    otherwise say "[The noun] does not reply." instead.

Section 3 - The Scenario

The Guard House is a room. "Here you spend all your nights. Bullet-proof windows offer a panoramic view of serene cliffs, palm trees, and a moonlit ocean. Occasionally someone is foolish enough to try a cliff ascent or even an attack by helicopter, but lately things have been pretty quiet.

```
The mansion is up the hill behind you, security lights
ablaze."
```

```
The grey telephone is a telephone in the Guard House.
Understand "6885" as the grey telephone. "Before you
is a grey telephone. In black marker someone has
written on it: MAIN OFFICE 2802."
```

```
Before going a direction in the Guard House, say "And
leave your post? The boss would have you flayed. No
kidding." instead.
```

In a game where the player could walk around, we would of course want to add a before rule so that he automatically hung up any phone he was using before leaving the room.

```
The Main Office is a room. The boss is a known woman
in the Main Office. A telephone called the red telephone
is in the Main Office. Understand "2802" as the red
telephone.
```

```
The Guild is a room. The ninja is an unknown man in
the Guild. A telephone called the black telephone is in
the Guild. Understand "4431" as the black telephone.
```

```
Potter's Pizza is a room. The pizza delivery boy is a
known man in Pizza. A telephone called the saucy
telephone is in Pizza. Understand "8885" as the saucy
telephone.
```

```
The chatter of the boss is the Table of Boss
Conversation. The chatter of the delivery boy is the
Table of Pizza Conversation. The chatter of the ninja is
the Table of Ninja Conversation.
```

After calling the red telephone on something for the first time:

say "'Yes?' asks the boss. Her voice is especially husky this evening. Maybe that night of passion isn't so far off after all."

Table of Boss Conversation

topic	reply
"love, sex, what's right/wrong with this picture"	"Some sort of night/night" "Oh. Maybe she'd go for some pizza, though."
"pizza"	"I have some. No pepperoni, though,' she says, sounding dreamy. Yes, this is definitely time for a call to your old friend, the pizza boy."
"I'm on my way"	"Don't worry about it,' says the boss crisply. 'I have everything under control.'"

Table of Pizza Conversation

topic	reply
"pizza"	"Pepperoni special tonight!' he says proudly."
"pepperoni"	"Pepperoni is included free on ALL our pizzas,' he says proudly."
"no pepperoni"	"Well, I don't see why you'd want that,' replies the boy sniffily. 'It's free!'"
"jalapeno"	"Sorry, we're out of jalapenos this evening. There was a run on them.'"
"sausage"	"Sausage, sure, we can do you sausage.'"
"bacon"	"There's currently an embargo on Canadian pig products.'"
"cheese"	"We do four kinds,' says the boy, then lowers his voice confidentially. 'Actually, two of them are the same. Nobody ever counts. The stringy one, the one that comes in dollops and the orangey one. You know.'"
"pineapple"	"We can put pineapple on there, sure,' says the delivery boy, in a tone that lets you know his opinion of people who order fruit-based pizzas."
"pineapple and garlic"	"What kind of combination is that?' demands the delivery boy, finally losing all self-control."
"delivery"	"Well, I don't know,' says the boy in a worried voice. 'Last time I came there were attack dogs. And ninjas.'"
"tip"	"There's no need for a big tip you don't live to spend,' says the delivery boy quite firmly.""

Table of Ninja Conversation

reply
"Yes, still on for tonight,' confirms the voice at the other end of the line."
"Pizza delivery boys"

Test me with "call 2802 / examine boss / ask boss about night of passion / ask boss about pizza / listen to telephone / call delivery boy on telephone / ask boy about cheese / tell boy about no pepperoni / ask boy about delivery / tell boy about massive gratuity / attack boy".

...and it more or less writes itself from there.

---

253 **ExampleAlias** *WI*

A telephone with phone numbers of the standard American seven-digit length.

Seven-digit telephone numbers are too long for Inform to handle when compiling to the Z-Machine, but they will work under Glulx. To have this example succeed, make sure that you have selected the Glulx option in your settings menu.

"Alias"

A telephone is a kind of thing. Understand "phone" or "telephone" as a telephone.

A phone number is a kind of value. 999-9999 specifies a phone number.

Now we borrow some techniques from the Understanding chapter to set up dialing actions:

```
Understand "dial [phone number] on [telephone]" as
dialing it on. Understand "dial [phone number] on
[something]" as dialing it on.

Understand the commands "phone" or "telephone" or
"call" as "dial".

Understand "call [text]" or "phone [text]" or "dial [text]" or
"telephone [text]" as a mistake ("That's not a number
you know.").

Dialing it on is an action applying to one phone number
and one thing.

Report dialing it on:
    say "You dial [the phone number understood]."
```

This much is enough to let us dial telephone numbers and have Inform report that we've done so; it doesn't actually provide a telephone system such that we could reach and converse with other characters (but see the other telephone examples in the recipe book for more on how one might do that).

We'll set up a little political espionage scenario from which our player can make calls:

```
The Senator's Junior Suite is a room. "The Senator
appears, unfortunately, to have very precise habits: little
in the room has been moved from its usual place; the
trash can is empty; the bed has been remade[if the blue
paper is unexamined]. There may in fact be nothing to
find here[end if]."
```

```
The bed is an enterable scenery supporter in the Junior
Suite.
```

```
The player is wearing a housekeeping uniform and a
brunette wig. The player carries a telephone called a
Nokia.
```

Borrowing again from the chapter on Understanding, we might arrange things so that the player knows and can call a few standard numbers with such syntax as CALL HOME:

```
Understand "home" as 555-9200.
```

And what if we'd like to have the player learn some phone numbers during the game?

```
A thing can be examined or unexamined. Carry out
examining something: now the noun is examined.
```

```
Understand "Stephen" as 555-2513 when the blue
paper is examined.
```

This will understand CALL STEPHEN once the paper is examined; before that, the player will just get the "That's not a number you know" response that Inform uses for all attempts to call unknown names.

We'd better plant this paper for the player to find:

```
The blue paper is in the drawer. The description of the
blue paper is "It reads: 'Call Stephen - 555-2513'."
```

```
The drawer is part of the dresser. It is closed and
openable. The dresser is in The Senator's Junior Suite.
The lamp is on the dresser. The description of the
```

dresser is "The single drawer is [if the drawer is open]open[otherwise]shut[end if]."

Test me with "dial 555-9999 / call home on the telephone / phone the president / call stephen / open drawer / read paper / call stephen / put phone in drawer / close drawer / call stephen".

---

180 **Example** **Witnessed 2** *WI*

A piece of ghost-hunting equipment that responds depending on whether or not the meter is on and a ghost is visible or touchable from the current location.

"Witnessed"

The player carries a device called a Trifield Natural EMF Meter. The description of the Meter is "This cost a pretty penny off the internet, but it's worth it: according to the website it has been programmed by PhD physicists to ignore manmade sources of fields and to respond only to paranormal EMF changes.

It also features an optional Tone Alarm, which can be turned on to indicate when readings spike. If the alarm is off, the meter just reads out the magnetic and electric field levels on a scale from 0-100 microteslas, or 0-1000 V/m.

Since both fields are important, you keep the meter set to SUM mode. The meter has its own optional backlighting, so that you can see the reading even if your flashlight is off. Currently it is reading at [meter setting]." A Tone Alarm is part of the Meter. It is a device. The description of the Tone Alarm is "The Tone Alarm will make a noise, if the EMF picks up a spike."

To decide what number is meter setting:
    if the meter is switched off, decide on 0;
    if a ghost is touchable, decide on 35;
    if a ghost is visible, decide on 12;
    decide on 0.

After switching on the meter:
    say "You turn on the meter. The needle steadies at [meter setting]."

Every turn: if the meter setting is greater than 10 and the Tone Alarm is switched on, say "[The Tone Alarm] shrieks."

Thirtieth Street Station is a room. "A huge, high, rectangular room with coffered ceilings, which looks grand but mostly makes you feel lonely and small. There are long benches in rows down the middle of the room, and an information desk with the train times, and a series of ticket windows, none of which matters very much at the moment."

The benches are an enterable supporter. They are scenery in the Station. The information desk is scenery in the Station. Some ticket windows are scenery in the Station. Instead of examining scenery in the Station: say "You're fairly sure that whatever is going on here has nothing to do with [the noun]." Understand "window" as ticket windows.

The mural is fixed in place in Thirtieth Street. "At the north side of the station is a particularly pointless and empty annex to the main room. It is dominated by a huge relief of sorts, and this is what you remember." Understand "metal" or "relief" or "huge" as the mural. The description of the mural is "It is both stylized and confusing, but you think it might be supposed to

represent the various tasks and occupations of Philadelphia's population. The portions closer to the ground look as though they have recently been subjected to a light dusting of talcum powder. No unusual prints are evident."

The wind chimes are fixed in place in Thirtieth Street. "Carefully attached to the wall with a piece of duct tape and a hook is a light-weight set of wind chimes. Someone else has been here before you, it seems." The description is "Several of your friends use wind chimes as a sort of ghost alarm, since ghosts sometimes cause very localized movements of air when there is no natural breeze."

A ghost is a kind of person. The pale figure is a ghost.

At 9:03 AM: move the pale figure to the location; say "You shiver with some sort of presence."

Test me with "turn on alarm / turn on meter / z / z / z / x figure".

---

299 **Example** **Tom's Midnight Garden** *WI*

A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running. Time on the face is also reported differently depending on whether the clock is analog or digital.

Time can also be understood as a token, and the time parsed will be recorded as "the time understood". So therefore, if we wish for clocks which may be set:

"Tom's Midnight Garden"

A clock is a kind of device. A clock has a time called the current time. A clock can be analog or digital. The current time of a clock is usually 9:01 AM. The description of a clock is "It shows the time to be [if analog]about [the current time to the nearest five minutes in words][otherwise][the current time][end if]."

Understand "set [clock] to [time]" as setting it by time. Setting it by time is an action applying to one thing and one time.

Instead of setting a clock to something:
    say "[The noun] can be set only to a time of day, such as 8:00 AM, or midnight."

Carry out setting a clock by time:
    now the current time of the noun is the time understood.

Report setting a clock by time:
    say "You set [the noun] to [time understood]."

Every turn:
    repeat with item running through switched on clocks:
        now the current time of the item is one minute after the current time of the item.

The Hall is a room. The grandfather clock is a fixed in place analog clock in the Hall. The travel clock is a switched on digital clock in the Hall. When play begins: now the right hand status line is "[time of day]".

Test me with "examine grandfather clock / set it to midnight / switch it on / wait / wait / wait / examine it / set travel clock to 4:12 / examine it".

ExampleGinger Beer

A portable magic telescope which allows the player to view items in another room of his choice.

Suppose we want to have a pair of linked lenses so that the player can look into one of them and see things which occur in room containing the other lense.

We begin simply with a bit of environment for the player to wander around:

"Ginger Beer"

The Ginger Beer Factory is a room. "In the center of the room is an enormous pot filled with crushed ginger, which seems to be bubbling slightly on its own. The fumes are overwhelming."

The pot is scenery in the Ginger Beer Factory. The description of the pot is "Cast iron." In the pot is a bubbling brew.

Instead of smelling the Ginger Beer Factory: try smelling the brew.

Instead of smelling the brew, say "You blink back tears."

The Storeroom is south of the Ginger Beer Factory. "The walls here are lined with a prodigious number of small, rounded bottles, each with a screw top and a smiling pirate on the label."

The Clippings Room is west of the Ginger Beer Factory. "A clean room lined with steel tables, for preparing ingredients."

Some steel tables are a supporter in the Clippings Room. They are scenery. The description is "They are roughly the size and height of laboratory worksurfaces."

The quantity of dandelion is on the steel tables. The description is "Horrible common weed."

The wooden box is on the steel tables. It is openable and closed. The description is "A large wooden box with a lid, used for ingredient storage. There is a label on the lid."

The label is part of the box. The description is "BURDOCK: the root beaten with a little salt and laid on the place suddenly easeth the pain thereof, and helpeth those that are bit by a mad dog:... the seed being drunk in wine 40 days together doth wonderfully help the sciatica: the leaves bruised with the white of an egg and applied to any place burnt with fire, taketh out the fire, gives sudden ease and heals it up afterwards.... The root may be preserved with sugar for consumption, stone and the lax."

The quantity of burdock is in the box. The description is "It looks like a kind of thistle."

```
Some bottles are in the Storeroom. They are scenery.
The description is "They are smaller than the average
bottle, because more potent." Instead of taking the
bottles, say "Take one away and the whole lineup will
cascade to the floor."
```

Now for the lenses themselves:

```
A lense is a kind of thing.
```

```
The large end of the telescope is a lense in the Ginger
Beer Factory. "There is a large glass lense propped
against the wall, in which are reflected all the contents
of the room." Understand "glass" or "lense" as the large
end.
```

```
The small end of the telescope is a lense in the
Storeroom. "There is a small glass lense sitting on the
floor. Due to some curious effect of the optics, it
appears to be giving a view of somewhere else entirely."
Understand "glass" or "lense" as the small end. The
description is "A gleaming lense about the size of a
pound coin."
```

Here is the critical bit, which needs to be somewhat flexible, since the large end of the telescope could in theory be left anywhere in the game (and should still work).

```
After deciding the scope of the player while the small
end is carried by the player:
    let there be the holder of the large end;
    place there in scope.
```

```
Before searching the small end when the small end is
not carried by the player:
    say "(first picking up [the small end] and holding it to
your eye)";
    silently try taking the small end.
```

```
Instead of searching the small end when the player is
not carrying the small end:
    say "It's too hard to look through the small end from a
distance."

Instead of searching the large end,
    say "You see only your own reflection."
```

We also want to make sure that the player who looks through the small lense does not see the large lense listed among the contents of the other location:

```
Definition: a thing is recognizable if it is not a lense.

Instead of searching the small end:
    let the far side be the holder of the large end of the
telescope;
    say "You peer into the little lense and through it see,
in [the far side], [the list of recognizable things in the far
side]."

Test me with "examine lense / south / examine lense /
look through lense / north / look through small lense".
```

And we're done.

A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

We start out by giving ourselves a capacious recording device:

"If It Hadn't Been For..."

```
The digital recorder is a device. The description is "A
noise-activated recorder, which time-stamps each
recording segment. It has space for about 60 short
recordings."


Every turn:
    if the digital recorder is switched on and the number
of blank rows in the Table of Recorded Content is 0,
now the recorder is switched off.


Table of Recorded Content

sound time stamp
action time text
with 60 blank rows.
```

And most of what follows is attaching sounds to various events. (We could have made noises associated with all the actions, but for simplicity we stuck to a few.)

The thing to note here is that the recording happens as part of Carry out, not as part of Report, so sounds will be recorded even when they are the result of non-player action when the player is not even in the room.

```
Carry out opening something in the presence of the
switched on recorder:
    record "A creaking noise, as of something being
opened."


Carry out someone opening something when the
switched on recorder can see the noun:
    record "A creaking noise, as of something being
opened."
```

```
Carry out closing something in the presence of the
switched on recorder:
    record "A creaking followed by a slam."

Carry out someone closing something in the presence
of the switched on recorder:
    record "A creaking followed by a slam."

Carry out someone going to a room (called destination)
in the presence of the switched on recorder:
    if the destination is the holder of the recorder, record
"Footsteps, growing louder.";
    otherwise record "Footsteps, fading out."

Carry out going to a room (called destination) in the
presence of the switched on recorder:
    if the destination is the holder of the recorder, record
"Footsteps, growing louder.";
    otherwise record "Footsteps, fading out."

Carry out someone eating something in the presence of
the switched on recorder:
    record "Loud uncouth chewing sounds."

Carry out eating something in the presence of the
switched on recorder:
    record "Distant muffled chewing sounds."

To record (noise - some text):
    if the number of blank rows in the Table of Recorded
Content is 0, rule succeeds;
    choose a blank row in the Table of Recorded
Content;
    now time stamp entry is the time of day;
    now sound entry is noise.
```

```
Understand "play [something]" as listening.


Instead of listening to the recorder:
    if the number of filled rows in the Table of Recorded
Content is 0, say "The recorder remains blank." instead;
    repeat through the Table of Recorded Content:
        say "[line break][time stamp entry]: [sound entry]";
    say paragraph break.


The Haunted House is a room. The squeaky cupboard
is an openable enterable closed fixed in place container
in the House. The ghost is a man in the cupboard. The
Lawn is outside from the Haunted House.


Instead of opening the closed cupboard when the ghost
is in the cupboard: say "The cupboard stubbornly
refuses to open."


Every turn when the player is not in the House:
    if the ghost is in the cupboard:
        try the ghost exiting;
    otherwise if the cupboard is open:
        try the ghost closing the cupboard.


Before someone exiting when the person asked is in a
closed container (called the trap):
    try the person asked opening the trap.


Before someone entering a closed container: try the
person asked opening the noun.


Before going to the House when the House contains the
```

```
ghost:
    try the ghost entering the cupboard;
    try the ghost closing the cupboard.
```

```
The player carries the recorder, chips, and a sandwich.
The sandwich is edible. The chips are edible.
```

```
Carry out someone eating the chips in the presence of
the switched on recorder: record "An incredible racket of
a packet being opened." Carry out eating the chips in
the presence of the switched on recorder: record "An
incredible racket of a packet being opened."
```

```
Test me with "open cupboard / drop recorder / switch it
on / eat chips / out / wait / wait / wait / in / switch
recorder off / play recorder".
```

Now we're at liberty to record evidence of the ghost getting out of the cupboard and getting back in, while we ourselves stand about on the lawn.

Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

We rely here on the understanding-by-relations rules we've already learned, but there is an additional trick: we want to make sure that if the player types "original" or "actual", this word will not be taken to refer to the thing modeled:

```
"Originals"
```

```
A model is a kind of thing. 100 models are in the model-
```

```
repository.

Appearance relates one thing to various models. The
verb to be shown by means the appearance relation.

Indication relates a model (called X) to a thing (called Y)
when Y is shown by X and Y is suitable.

Understand "actual" or "original" as "[actual]".
Understand "[actual]" as something when the item
described is not a model.

Definition: a thing is suitable:
    if the player's command includes "[actual]", no;
    yes.

Understand "[something related by indication]" as a
model.

After printing the name of a model (called target): say "
[random thing shown by the target]"
```

Now our duplication command -- for the sake of simplicity, we'll suppose that in this scenario the player is duplicating objects by magic rather than creating them out of physical materials or supplies:

```
Understand "duplicate [something]" as duplicating.
Duplicating is an action applying to one visible thing.

The duplicating action has an object called the selected
model.
```

```
Setting action variables for duplicating:
    let N be a random model in the model-repository;
    now the selected model is N.


Check duplicating:
    if the selected model is nothing, say "You're out of
power." instead.


Carry out duplicating:
    now the noun is shown by the selected model;
    move the selected model to the player.


Report duplicating:
    say "You concentrate and manifest [a selected
model]."
```

Now, the challenge is that we want to print the word "actual" before printing the name of an object, but only during disambiguation questions and only when we are not printing the name of the object as part of a model-name! (If we are not careful about the latter point, we will get responses such as "Which do you mean, the model actual deer or the actual deer?" which of course defeats the whole purpose.

The way around this is to remember that activities stack: we're printing the name of the deer while printing the name of a model that involves the deer. So if we set a flag while printing the name of a model, we can control the way the deer's name prints during the transaction. (We could use our ...while clause to specify while not printing the name of a model, except that we're already using it for "while asking which do you mean", and these do not stack.) So:

```
The virtual-context is a truth state that varies. The
virtual-context is false.


Before printing the name of a model:
    now virtual-context is true.
```

After printing the name of a model:
    now virtual-context is false.

Before printing the name of something (called target)
while asking which do you mean:
    if the target is not a model and virtual-context is false:
        say "actual ".

Forest is a room. It contains a deer and a daisy. The
deer is an animal.

Test me with "duplicate deer / x model deer / x deer
model / drop deer / x deer / actual / x deer / model".

Example **Mirror, Mirror** *WI*

The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

"Mirror, Mirror"

The Sorcerer's Workshop is a room. "The sorcerer's den is a dusty, whispering place. A grandfather clock with skeletal hands reads [the time of day in words]. The floorboards are stained where that porridge just wouldn't come out."

The Apprentice's Pantry is east of the Workshop. "This is where the aproned apprentice traditionally makes the camomile tea, cleans out the jackdaw cages and

furtively examines purloined artefacts."

When play begins: erase the mirror.

The player carries a magic mirror. The magic mirror has a text called the mirror vision.

To erase the mirror: now mirror vision of the mirror is "The mirror is polished clean, and has no impression upon it."

To say current room description: try looking.
To expose the mirror:
    say "The mirror shines momentarily with a dazzling light.[paragraph break]";
    now mirror vision of the mirror is the substituted form of "The hazy image in the mirror preserves a past sight: [line break][current room description]All is distorted and yet living, as though the past and present are coterminous in the mirror."

Understand "hold up [something preferably held]" or "hold [something preferably held] up" as holding aloft.
Holding aloft is an action applying to one carried thing.
Report holding aloft: say "You hold [the noun] aloft."

Instead of rubbing the mirror: erase the mirror; try examining the mirror. Instead of holding aloft the mirror: expose the mirror.

The description of the mirror is "[mirror vision of the mirror]".

Test me with "look / examine mirror / hold up mirror / z / look / x mirror / rub mirror / east / hold mirror up / west / x mirror".

---

221 **Example** **Actor's Studio** *WI*

A video camera that records actions performed in its presence, and plays them back with time-stamps.

Here we construct a video camera to track and play back actions:

"The Actor's Studio"

Section 1 - The Video Camera

The video camera is a thing carried by the player.

Table of Videotape

time-stamp	action
9:00 AM	waiting

with 25 blank rows.

Mode is a kind of value. The modes are idle, recording, and playing back. The video camera has a mode. The video camera is idle.

Understand "play back" as playing back. Instead of switching on the camera, try tuning the camera to recording. Instead of switching off the camera, try tuning the camera to idle.

The description of the video camera is "It is currently [mode]; its available settings are idle, recording, and playing back."

Understand "set [camera] to [a mode]" as tuning it to. Tuning it to is an action applying to one thing and one mode.

Instead of setting the camera to something:
    say "The available settings are idle, recording, and playing back."

Check tuning it to:
    if the noun is not the camera, say "Only the video camera can be set to [the mode understood]." instead.

Carry out tuning it to:
    now the mode of the noun is the mode understood.

Report tuning it to:
    say "You set [the noun] to [mode understood]."

After an actor doing something when the video camera is recording:
    if the current action is tuning the video camera to recording, make no decision;
    if the number of blank rows in the Table of Videotape is greater than zero:
        choose a blank row in the Table of Videotape;
        now the recorded action entry is the current action;
        now the time stamp entry is the time of day;
    otherwise:
        now the video camera is idle;
        say "The video camera runs out of recording

```
memory and switches off.";
    continue the action.



Every turn when the video camera is playing back:
    say "On the camera screen, you see [run paragraph
on]";
    let starting playback be false;
    repeat through the Table of Videotape:
        if the recorded action entry is not waiting:
            now starting playback is true;
            say "[line break] -- [if the actor part of the
recorded action entry is the player]you [end if][the
recorded action entry], time stamped at [time stamp
entry]";
            blank out the whole row;
    if starting playback is false, say "only static.";
    otherwise say paragraph break.



Section 2 - The Scenario



The Actor's Studio is a room. Lucas is a man in the
Actor's Studio. Persuasion rule: persuasion succeeds.



The Studio contains an edible thing called a croissant.



Test me with "set camera to recording / x lucas / lucas,
take inventory / lucas, eat croissant / set camera to
playing back / z".
```

Notice that both Lucas' implied taking action (picking up the croissant) and his eating action are recorded on the same move.

ExampleClaims Adjustment

An instant camera that spits out photographs of anything the player chooses to take a picture of.

We start by creating a camera and a photograph object. As usual when we want to have a kind of object that can be dispensed in bulk, we start off with a bunch of identical instances of the object out of play (in this case, kept in an out-of-play container called "film roll"); we can then move them into play and give them characteristics when they're needed.

Each photograph can depict exactly one thing -- we're assuming that the player is not a landscape photographer here -- so we create a relation to indicate what is shown by each photograph. We'll then use that relation to determine how photographs are described, named, and parsed:

```
"Claims Adjustment"

A photograph is a kind of thing. 36 photographs are in
the film roll.

Appearance relates one thing to various photographs.
The verb to be shown by means the appearance
relation.

The description of a photograph is usually "It shows [a
random thing which is shown by the item described]."

Understand "of [something related by reversed
appearance]" as a photograph.
```

This allows the player to refer to any photograph by its subject: useful if we have a large number of them.

Now we create an action to let the player use the camera and generate these photograph objects:

```
The player carries a cheap instant camera.
```

```
Understand "photograph [something] with [camera]" as
photographing. Understand "photograph [something]
with [something preferably held]" as photographing.
Photographing is an action applying to one visible thing
and one carried thing, requiring light.


The photographing action has an object called the
selected film.


Setting action variables for photographing:
    let N be a random photograph in the film roll;
    now the selected film is N.


Check photographing:
    if the second noun is not the camera, say "You need
a camera for that purpose." instead.


Check photographing:
    if the noun is the camera, say "Sadly impossible."
instead.


Check photographing:
    if the selected film is nothing, say "You're out of film."
instead.


Carry out photographing:
    now the noun is shown by the selected film;
    move the selected film to the player.


Report photographing:
    say "Your camera instantly spits out [a selected film]."
```

Now we use two activities from the Activities chapter to describe the photographs to the player more elegantly:

```
After printing the name of a photograph (called target):
    say " of [a random thing which is shown by the
target]".


After printing the plural name of a photograph (called
target):
    let N be the holder of the target;
    say " of [a list of things which are shown by
photographs which are held by N]";
    if the number of things which are shown by
photographs which are held by N is greater than one,
say " (variously)".
```

And finally we provide a brief scenario to give the player something to take pictures of:

```
The Treasure Room is a room. "Despite the fancy
name, this is no more than a closet -- albeit a closet with
its own special circuit on the house alarm."


The Treasure Room contains a small Degas, a Ming
vase, and a collection of South African krugerrands. The
player is carrying insurance forms, a first-class stamp,
and a security envelope.


The description of the forms is "Completely filled out in
black ink in block letters: now all you need to do is
attach photographic evidence of the objects you wish to
insure."


Test me with "photograph degas / i / photograph degas /
i / x photograph of degas / photograph me / x
photograph of me / i / photograph vase / photograph
camera / photograph collection / g / i / test more".
```

Test more with "x photograph of collection / x photograph of krugerrands / x photograph of collection of south african krugerrands / photograph photograph of degas / x photograph of photograph of degas".

Chapter 10: Physics: Substances, Ropes, Energy and Weight

er Damage-Prone Substances; *§10.5. Volume, Height, Weight*; *§10.6. Ropes*; *§10.7. Electricity and Ma*

Contents of *The Inform Recipe Book*
Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics
Chapter 11: Out Of World Actions and Effects
Indexes of the examples

§10.1. Gases

Inform normally assumes that everything is solid. It has no built-in support for gases or liquids, because they have so many different behaviours. For instance, is the important thing about gas that it diffuses, or that we breathe it, or that it mixes with other gases to react, or that it sometimes obscures vision? The answer depends on what we are trying to write, and for this reason Inform leaves it up to us.

Gases are easier to deal with than liquids, because they tend to be everywhere in their location at once (unlike a liquid which might form a pool on the floor) and because they diffuse and mix by themselves (rather than being carried around or brought into contact with each other by the player). On the other hand, unlike liquids, gases are compressible: they can be present at low or high pressures, that is, in low or high concentrations.

The simplest approach is the one in **Only You...**, where rooms are either filled with smoke or else smoke-free. Smoke gradually fills through the map, obscuring vision: no attempt is made to conserve the total quantity of smoke, as we assume that some fire is churning it out continuously to replace what diffuses away.

Lethal Concentration 1 and **2** simulate a gas diffusing through a three-dimensional maze of rooms, and becoming dangerous above certain concentrations. There is just one possible gas, and it is modelled by giving each room a number which represents the concentration (in parts per million). This enables us to conserve the total amount of gas, or to have it released or captured by sources and sinks of given capacity.

This could be extended by giving each room similar concentration levels for other gases, and providing for the diffusion rule to notice when different gases come into contact; or by giving a concentration (and also, for realism, a volume) to each closed

container, applying rules for capturing and releasing gases as containers are opened and closed.

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics: §9.12. Cameras and Recording Devices

Onward to §10.2. Liquids

Example 255: ★★ **Lethal Concentration 1** A poisonous gas that spreads from room to room, incapacitating or killing the player when it reaches sufficient levels.

Example 74: ★★★ **Only You...** Smoke which spreads through the rooms of the map, but only every other turn.

Example 257: ★★★ **Lethal Concentration 2** Poisonous gas again, only this time it sinks.

§10.2. Liquids

Liquids are notoriously difficult to simulate well. A fully thorough approach consumes endless storage and can be very finicky to write and keep realistic. It is essential to decide what aspect of a liquid's behaviour is actually needed in a given story, and to simulate only that. For instance, if we only need a little chemistry, where a player can add (say) water to salt and make a solution, we do not want to fool around with calculating quantities or concentrations: what's important is that "some water" (amount unspecified) combines with "some salt" to produce "some salty water". We should no more calculate precisely here than we would work out where all the furniture is to the nearest inch. Good advice for handling liquids is to simulate the least amount of realism possible, but no less.

Sometimes all we want is a down-in-one drink: we needn't simulate the actual liquid, just the bottle it comes in, and all we need is to handle the "drinking" action. See

Beverage Service, and also **3 AM**, where carbonated drinks can be shaken - again simulating the vessel, not the liquid.

Some elementary biochemistry in **Xylan** is done simply by... well, the point is that two different liquids are represented by single things each, and a chemical reaction simply switches one for the other.

In **Frizz**, we allow any container to be filled with water (only) and we simulate what happens to any solid objects also inside: some waterproof, some not. **Flotation** provides a well (always full of water), with rules to determine whether things dropped into it should sink or float.

Next we move up to quantitative approaches, where we remember not just whether a liquid is present, but how much of it. In its simplest form, we could have a drinking vessel from which we draw in sips, so that it can be full, half-empty or empty: see **Thirst**.

The example with the best compromise between simulation quality and complexity is **Lemonade**. Here we provide a kind of container called a "fluid container", not just a single cup, and each such vessel has a given "fluid capacity". Each holds only a single liquid at a time (so no mixtures) and can be empty or full to any level (rounded off to the nearest 0.1 fl oz). We can fill one vessel from another (unless it would make a mixture). But liquids leaving these vessels must be consumed - drunk or poured away without trace: we cannot make pools on the floor, or carry liquids in our cupped hands. There is no object representing "lemonade": there are only fluid containers, but which can be called LEMONADE if that is what they now contain.

Savannah is a light elaboration of Lemonade, showing how liquids might be poured on other objects, as for instance to extinguish a fire.

Noisy Cricket extends **Lemonade** to allow for mixing, though then the number of different possible mixtures is so large that complexity increases greatly. **Lakeside Living** extends **Lemonade** differently to add a "liquid source" kind, a form of fluid container which has infinite fluid capacity and is scenery - ideal for a lake, river or spring.

See **Bags, Bottles, Boxes and Safes** *for stoppered bottles which could also be used for carrying liquids around in*

See **Heat** *for keeping liquids warm in insulated containers*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.1. Gases

Onward to §10.3. Dispensers and Supplies of Small Objects

Example 117: ★ **Thirst** A waterskin that is depleted as the player drinks from it.

Example 122: ★ **Beverage Service** A potion that the player can drink.

Example 403: ★ **Flotation** Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

Example 288: ★★ **Xylan** Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

Example 192: ★★★ **Frizz** Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

Example 193: ★★★ **3 AM** A shake command which agitates soda and makes items thump around in boxes.

Example 266: ★★★ **Lemonade** Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

Example 267: ★★★ **Savannah** Using the liquid implementation demonstrated in Lemonade for putting out fires.

Example 276: ★★★ **Noisy Cricket** Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

Example 333: ★★★ **Lakeside Living** Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

---

§10.3. Dispensers and Supplies of Small Objects

A slightly tricky situation arises in IF when we want to offer the player a simulation of a near-infinite supply of something: a napkin dispenser from which he can keep taking more napkins, or an infinite selection of pebbles on a beach, or something of that nature.

One approach is simply to limit the number of items the player is allowed to pick up at a time, while maintaining the fiction that there are more of these items in existence than the player is allowed to interact with. **Extra Supplies** demonstrates this.

The task becomes harder if we do want to let the player have as many napkins as he wants. In some languages, it is possible to generate new objects on the fly after the story has begun (called "dynamic object creation"), and something like this is possible if we are compiling for Glulx. (See the Inform extensions site for examples.) Usually, though, it is less complicated and almost as effective simply to have a very large supply of existing objects, which are moved in and out of play as the player needs them. **Pizza Prince** demonstrates how to do this with slices of pizza.

See **Ropes** *for an example involving divisible pieces of string, which relies on similar techniques*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.2. Liquids

Onward to §10.4. Glass and Other Damage-Prone Substances

Example 378: ★ **Pizza Prince** Providing a pizza buffet from which the player can take as many pieces as he wants.

Example 124: ★★ **Extra Supplies** A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

---

§10.4. Glass and Other Damage-Prone Substances

Just as Inform normally assumes everything is solid, it also assumes that these solid objects will not buckle, crack, break or deform under pressure, and cannot be fragmented. But breakability adds realism, and breakage need not be negative: sometimes we want the player to break her way in to something.

In IF the word DROP is more often used to mean "put down" or "leave behind" than "let go from a height": so it is perhaps unfair that in **Ming Vase** something fragile, when dropped, shatters (into nothing). In **Spring Cleaning**, fragile objects must be explicitly attacked by the player in order to break, and although they leave no tangible debris behind, their loss is at least remembered. **Kyoto** provides a general-purpose model for things being thrown at other things, with consequences including things moving (even between rooms) as well as breaking each other.

Debris from breakages is to be avoided if possible because it means keeping track of increasing numbers of objects. But we can increase realism by allowing something to have a visibly "broken" state, which it changes to when it breaks, rather than simply vanishing. **Terracottissima** provides for broken and unbroken flowerpots in this way.

Since "part of" allows us to have two objects joined together into what the player sees as one, it also gives us a natural seam which allows the whole to be broken back down into its component parts, and this is the neatest way of providing a breakage into pieces. **Paddington** demonstrates a cutting action which allows component parts to be cut away from their holders but will only make small surface gashes in any individual thing: so the player can cut something up, but only into the pieces we specifically choose to allow. Cutting also forces an opening into containers.

★ *See* **Combat and Death** *for a robot that breaks into its component limbs when shot with a blaster*

See **Goal-Seeking Characters** *for a character who eats donuts, leaving crumbs on the floor*

See **Volume, Height, Weight** *for containers breaking under the weight of their contents*

See **Ropes** *for cutting up string into up to 35 different pieces of different lengths - a limit the player is unlikely to find out about, but a limit all the same, and an expensive solution since we need 35 different things for the "debris" when string is "broken"*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.3. Dispensers and Supplies of Small Objects

Onward to §10.5. Volume, Height, Weight

Example 97: **Ming Vase** ATTACK or DROP break and remove fragile items from play.

Example 123: **Spring Cleaning** A character who sulks over objects that the player has broken (and which are now off-stage).

Example 195: **Paddington** A CUT [something] WITH [something] command which acts differently on different types of objects.

Example 314: **Terracottissima** The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

Example 404: **Kyoto**

Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

§10.5. Volume, Height, Weight

What should fit into what? Inform has basically three sizes: small, person-sized, and room-sized. The difference between "small" and "person-sized" doesn't appear much, but it's the difference between an ordinary container and an enterable container; the fact that a person cannot get inside an ordinary container is one of the few size-related rules built into Inform. It will not object to, say, a fishing rod being put inside a matchbox.

Inform does have one built-in measure of the size of a container: its "carrying capacity". This is a maximum number of contents:

```
The carrying capacity of the rucksack is 3.
```

This of course allows three anvils, while forbidding four postage stamps. To do better, we need units of measurement, and **Dimensions** demonstrates setting these up. **The Speed of Thought**, meanwhile, ventures into the area of unit conversion: having multiple types of unit and being able to express them to the player, or parse these in the player's input.

To be fully realistic in what will fit into what, we need sophisticated three-dimensional models of shapes, both of the items being carried and of the free space remaining inside containers. **Depth** elegantly simplifies this by approximating items as cuboids, with a given width, length and height: these multiply to give a volume. To fit in a container, a new item's volume must not exceed the volume remaining inside the container, and in addition its three dimensions must also fit in one of the possible arrangements at right angles to the sides. (So this system would indeed prevent a 1x1x100 fishing rod from being put inside a 5x2x1 matchbox, but would also prevent a 12x1x1 pencil from being put into a 10x10x1 box, because it would need to be turned diagonally to fit.)

Lead Cuts Paper provides a different constraint: here we do not let light-weight containers hold heavy objects.

Weight comes in a different way into **Swerve left? Swerve right? Or think about it and die?**, which exploits up/down map connections to work out which way gravity would take a rolling marble.

See **Liquids** *for containers with liquid capacity*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight
Back to §10.4. Glass and Other Damage-Prone Substances
Onward to §10.6. Ropes

Example 232: ★ **Swerve left? Swerve right? Or think about it and die?** Building a marble chute track in which a dropped marble will automatically roll downhill.

Example 268: ★ **Depth** Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

Example 258: ★★ **Dimensions** This example draws together the previous snippets into a working implementation of the weighbridge.

Example 270: ★★ **The Speed of Thought** Describing scientifically-measured objects in units more familiar to the casual audience.

Example 259: ★★★ **Lead Cuts Paper** To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster.

§10.6. Ropes

Ropes, chains and similar long, thin, bendable items present three problems: they are like a liquid in that (unless unbreakable) they can be divided arbitrarily into smaller and smaller portions of themselves, they can be in two or more places at once (even in two or more rooms at once), and they can be tied down at either or both ends, allowing them to occupy an uneasy state in between being "portable" and "fixed in place". Even when all this is simulated, they allow us to pull on one end and so to exert force at the other - allowing action-at-a-distance which Inform's realism rules would ordinarily forbid. Ropes are hard. And it is very difficult to imagine everything a player might try when given a fully practical rope with which to solve puzzles.

Snip solves the divisibility question, allowing string to be cut or retied into lengths of any size, with all the consequences for describing and understanding that this entails.

Otranto provides a lengthy but straightforward approach to the other rope-related issues, subject to the simplifying assumptions that a rope is indivisible, has about the length of the distance between two adjacent rooms, and cannot be tied to another rope.

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.5. Volume, Height, Weight

Onward to §10.7. Electricity and Magnetism

Example 228: ★★★ **Otranto** A kind of rope which can be tied to objects and used to anchor the player or drag items from room to room.

Example 261: ★★★ **Snip** A string which can be cut into arbitrary lengths, and then tied back together.

---

§10.7. Electricity and Magnetism

Electrons are so tiny, and move so fast, that we will never want to simulate them in ordinary IF. So we simply regard electricity and magnetism as behaviours which are either present or not present, and which have instantaneous effects.

In **Witnessed 1**, batteries provide electricity to enable a "device" to work. Even if switched on, a device with no battery will be ineffective.

Larger voltages are exposed in **Electrified**, which makes certain items untouchable, and ensures that an experienced electrician will not even try.

Rules of Attraction provides for a magnet which attracts metallic items just strongly enough to stick together until pulled apart for any reason.

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.6. Ropes

Onward to §10.8. Fire

Example 346: ★ **Rules of Attraction** A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

Example 408: ★ **Electrified** Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances.

Example 139: ★★★ **Witnessed 1** A kind of battery which can be put into different devices, and which will lose power after extended use.

---

§10.8. Fire

Fire exhibits some of the properties of a gas: it is only vaguely located and tends to spread out, though it passes by touch rather than on the air. It is hazardous to life, through direct contact, heat, and smoke. Better governed, it provides light and warmth. Worse governed, it consumes almost anything it comes into contact with.

Here the problem with "debris" is not so much that we need potentially hundreds of new objects to represent broken items: instead, fire could sweep through a work of IF destroying so much that no play is possible any longer. Setting up a problem in which the player must defeat a fully-capable fire is difficult to balance.

As with liquids, it is best to simulate the least amount of fire that the design will allow. **Bruneseau's Journey** provides a single candle which can be lit, or blown out, but where fire can never transfer from the candle's end to anything else - or vice versa: the player's source of fire, with which to light the candle, is discreetly neglected.

In the more realistic **Thirst 2**, a campfire is lit using a tinderbox, so that fire does transfer from one thing (tinder) to another (the campfire): but it is always confined to just these two items, and can be used only for light and warmth.

The Cow Exonerated provides a class of matches that can light any flammable object, but assumes that burning objects requires only one turn; lighting one thing does not burn another.

In Fire or in Flood provides a complete simulation of what we might call "wild-fire": combustion which spreads through arbitrary objects and rooms, destroying all in its path.

See **Examining** *for a way to describe objects as charred once they have been partly burnt*

See **Heat** *for one consequence of fire having touched something*

See **Gases** *for an implementation of smoke without fire, if this can exist*

See **Liquids** *for water being used to extinguish a simple fire*

See **Lighting** *for other uses of candles and torches as light sources*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.7. Electricity and Magnetism

Onward to §10.9. Heat

Example 118: **Thirst 2** A campfire added to the camp site, which can be lit using tinder.

Example 150: **Bruneseau's Journey** A candle which reacts

to lighting and blowing actions differently depending on whether it has already been lit once.

Example 418: ★★ **The Cow Exonerated** Creating a class of matches that burn for a time and then go out, with elegant reporting when several matches go out at once.

Example 401: ★★★ **In Fire or in Flood** A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

---

§10.9. Heat

Since we prefer not to simulate burnt skin, and it is unsporting to kill a player outright merely for touching a hot object, heat is mostly used as a reason why something cannot be picked up at a given moment. This very basic puzzle is demonstrated in **Grilling**.

With the naked eye, it is not always easy to detect what is too hot to touch - a point made in both **Masochism Deli**, where the only solution is to keep picking up potatoes until one doesn't burn, and in **Hot Glass Looks Like Cold Glass**, where infrared goggles turn the scales.

If a hot object is not to be touched, will it stay hot forever? It might well, if it is a steak on an electric grill, but not if it is a recently-baked apple pie sitting on a window-sill. **Entropy** simulates the gradual return of temperature to equilibrium.

See **Electricity and Magnetism** *for items which shouldn't be touched because they are hot in a different way*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight
Back to §10.8. Fire

Onward to §10.10. Magic (Breaking the Laws of Physics)

Example 84: ★ **Grilling** A grill, from which the player is not allowed to take anything lest he burn himself.

Example 182: ★ **Entropy** All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

Example 326: ★ **Masochism Deli** Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

Example 331: ★★★ **Hot Glass Looks Like Cold Glass** Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

---

§10.10. Magic (Breaking the Laws of Physics)

Every previous section of this chapter has been about adding further realism constraints to Inform, to give it a more advanced understanding of physics (and chemistry). But Inform has plenty of physical laws built into it already, even if they are more mundane: inanimate objects do not spontaneously move or change, one solid object cannot pass through another, there are opaque barriers through which light cannot pass, and so on. These rules stop many attempted actions. (GO EAST - "The oak door is closed." GET EMERALD - "You can't, since the glass display case is in the way.")

In the real world, physics is not negotiable. ("Gravity: it's not just a good idea, it's the law.") But in stories, magic can sometimes happen, and in these examples some of the rules built into Inform are waived in special circumstances, which in effect cancel certain physical laws. Very many other magical tricks could be achieved: if you want

to make a given command work despite realism constraints, try typing ACTIONS - a testing command which reveals what is happening behind the scenes - and then typing the command you want. This should reveal which named rule is stopping it: you can then try suspending that rule, and seeing if the effect now works.

See **Magic Words** *for ways to create new single-word spell commands*

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight
Back to §10.9. Heat
Onward to §10.11. Mathematics

Example 215: **Magneto's Revenge** Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

Example 225: **Interrogation** A wand which, when waved, reveals the concealed items carried by people the player can see.

Example 227: **Transmutations** A machine that turns objects into other, similar objects.

Example 398: **Access All Areas** The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

---

§10.11. Mathematics

It is uncommon, but not absolutely unheard-of, to need to do detailed mathematical calculations in interactive fiction.

The Fibonacci Sequence demonstrates the calculation of a series of numbers; **Sieve of Eratosthenes** shows off how to calculate the prime numbers within a

certain range.

Number Study demonstrates conditional relations between numbers.

Start of Chapter 10: Physics: Substances, Ropes, Energy and Weight

Back to §10.10. Magic (Breaking the Laws of Physics)

Onward to Chapter 11: Out Of World Actions and Effects: §11.1. Start-Up Features

Example 240: **Number Study** The parity and joint magnitude relations explored.

Example 434: **The Fibonacci Sequence** The modest Leonardo Fibonacci of Pisa will be only too happy to construct his sequence on request, using an array.

Example 437: **Sieve of Eratosthenes** The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

Examples from Chapter 10: Physics: Substances, Ropes, Energy and Weight

Start of this chapter

Chapter 11: Out Of World Actions and Effects

Indexes of the examples

Example **Lethal Concentration 1**
A poisonous gas that spreads from room to room, incapacitating or killing the player when it

"Lethal Concentration"

A concentration is a kind of value. 200.9ppm specifies concentration. 200.9 ppm specifies concentration.

A room has a concentration called current concentration. A room has a concentration called former concentration.

Probability inverse is a number that varies. [This is expressed as an inverse of the actual probability of diffusion from one room to another, to avoid error.] Probability inverse is 20. [That is, any given molecule of gas has a 5% chance of leaving by a given doorway at any given minute. Probability inverse should never drop below 10, the maximum number of exits from the room.]

Every turn:
    follow the diffusion rules.

The diffusion rules are a rulebook.

A diffusion rule (this is the gas movement rule):
    repeat with space running through rooms:
        let sum be 0.0 ppm;
        repeat with way running through directions:
            let second space be the room way from the space;
            if second space is a room:

```
            let difference be the former concentration of
        the second space minus the former concentration of the
        space;
            increase sum by the difference;
        let sum be sum divided by probability inverse;
        now current concentration of the space is the
        former concentration of the space plus the sum.
```

A technical note: it would be possible to write "repeat with space running through rooms... repeat with second space running through rooms adjacent to the space" instead, but in practice this loops through all the rooms * all the rooms again * all the directions (to determine adjacency). Phrasing the loop this way omits the second multiplier. For a map of 25 rooms, this means that the loop runs 25 times faster than it would otherwise, and of course for a larger map the effect would be even more dramatic.

```
A diffusion rule (this is the resetting concentration rule):
    repeat with space running through rooms:
        now the former concentration of the space is the
    current concentration of the space.
```

```
The last diffusion rule (this is the lethal dosage rule):
    if the current concentration of the location is greater
    than LC50:
        say "The concentration in the air overpowers
    you...";
        end the story;
    otherwise:
        if the current concentration of the location is
    greater than TLV-STEL:
            say "You feel extremely uncomfortable in this
    environment."
```

```
Instead of doing something when the current
concentration of the location is greater than TLV-STEL:
    if going, continue the action;
    say "You can't work in this environment: your eyes
and nose sting and it hurts to breathe."
```

And, for testing purposes, a square grid of rooms:

Room 1A is west of Room 1B. Room 1B is west of Room 1C. Room 1C is west of Room 1D. Room 1D is west of Room 1E.

Room 2A is south of room 1A and west of Room 2B.
Room 2B is west of Room 2C and south of Room 1B.
Room 2C is west of Room 2D and south of Room 1C.
Room 2D is west of Room 2E and south of Room 1D.
Room 2E is south of Room 1E.

Room 3A is south of room 2A and west of Room 3B.
Room 3B is west of Room 3C and south of Room 2B.
Room 3C is west of Room 3D and south of Room 2C.
Room 3D is west of Room 3E and south of Room 2D.
Room 3E is south of Room 2E.

Room 4A is south of room 3A and west of Room 4B.
Room 4B is west of Room 4C and south of Room 3B.
Room 4C is west of Room 4D and south of Room 3C.
Room 4D is west of Room 4E and south of Room 3D.
Room 4E is south of Room 3E.

Room 5A is south of room 4A and west of Room 5B.
Room 5B is west of Room 5C and south of Room 4B.
Room 5C is west of Room 5D and south of Room 4C.
Room 5D is west of Room 5E and south of Room 4D.
Room 5E is south of Room 4E.

The former concentration of room 3C is 800.0 ppm.

For variety of testing, here is another room set-up, this time with some corridors and walls within; uncommenting it, and commenting out the connected grid, will let us explore what would happen in alternative cases, to prove to ourselves that the model works consistently.

[Room 1A is west of Room 1B. Room 1B is west of

```
Room 1C. Room 1C is west of Room 1D. Room 1D is
west of Room 1E.


Room 2A is west of Room 2B. Room 2B is west of
Room 2C. Room 2C is west of Room 2D. Room 2D is
west of Room 2E. Room 2E is south of Room 1E.


Room 3A is south of room 2A and west of Room 3B.
Room 3B is west of Room 3C. Room 3C is west of
Room 3D. Room 3D is west of Room 3E.


Room 4A is west of Room 4B. Room 4B is west of
Room 4C. Room 4C is west of Room 4D. Room 4D is
west of Room 4E. Room 4E is south of Room 3E.


Room 5A is south of room 4A and west of Room 5B.
Room 5B is west of Room 5C and south of Room 4B.
Room 5C is west of Room 5D and south of Room 4C.
Room 5D is west of Room 5E and south of Room 4D.
Room 5E is south of Room 4E.]
```

For the sake at least of seeing what's going on in the example, let's also provide the player with the means to view the gas diffusion graphically:

```
The status grid is a device carried by the player. The
status grid is switched on.


Every turn:
    try examining the grid.


Instead of examining the status grid:
    say "[fixed letter spacing][bar][line break]";
    say "|[state of room 1A]|[state of room 1B]|[state of
room 1C]|[state of room 1D]|[state of room 1E]|[line
```

```
break]";
    say "[bar][line break]";
    say "|[state of room 2A]|[state of room 2B]|[state of
room 2C]|[state of room 2D]|[state of room 2E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 3A]|[state of room 3B]|[state of
room 3C]|[state of room 3D]|[state of room 3E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 4A]|[state of room 4B]|[state of
room 4C]|[state of room 4D]|[state of room 4E]|[line
break]";
    say "[bar][line break]";
    say "|[state of room 5A]|[state of room 5B]|[state of
room 5C]|[state of room 5D]|[state of room 5E]|[line
break]";
    say "[bar][variable letter spacing][line break]".

To say bar:
    say "-----------------------------------------".

TLV is a concentration that varies. TLV is 30.0ppm.
[Long-term exposure maximum, safe for 8 hours a day.]

TLV-STEL is a concentration that varies. TLV-STEL is
50.0ppm. [Short-term exposure maximum, safe for
fifteen minutes max.]

TLV-C is a concentration that varies. TLV-C is
150.0ppm. [Absolute exposure ceiling.]

LC50 is a concentration that varies. LC50 is 300.0ppm.
[Concentration at which 50 percent of test subjects die
of exposure, usually expressed in terms of time and
body weight; in our LC50 these are factored in for the
player's weight for one minute.]
```

The values set for these would depend on the type of poisonous gas in

question; we'd want to adjust appropriately.

```
To say state of (space - a room):
    if the space is the location, say bold type;
    if current concentration of space is less than
10.0ppm, say " ";
    if current concentration of space is less than
100.0ppm, say " ";
    say current concentration of space;
    say roman type.
```

Now, in theory we might also want to account for sources and sinks, items that either inject poisonous gas into the environment or remove it again. For simplicity, we will assume that these contributions can also be calculated in ppm and that the total number of inert and poisonous gas molecules in a room never changes (so if poison gas molecules are added, an equal number of inert molecules are removed). If room pressure were able to change, our model would have to be improved, so let us assume for now that that never happens. We want this sink/source business to calculate before any other portion of the diffusion rulebook, so set it as a first diffusion rule.

```
A gas source is a kind of thing. A gas source has a
concentration called the contribution. The contribution of
a gas source is usually 30.0ppm.
```

```
Room 2B contains a gas source called a spigot. The
contribution of the spigot is 50.0ppm. Room 5A contains
a gas source.
```

```
A gas sink is a kind of thing. A gas sink has a
concentration called the contribution. The contribution of
a gas sink is usually 30.0ppm.
```

```
Room 5E contains a gas sink called a fan. The
contribution of the fan is 80.0ppm.
```

```
The first diffusion rule (this is the sources and sinks
rule):
    follow the sources rule;
    follow the sinks rule.



This is the sinks rule:
    repeat with item running through gas sinks:
        let space be the location of the item;
        decrease the former concentration of the space by
the contribution of the item;
        if the former concentration of the space is less than
0.0ppm, now the former concentration of the space is
0.0ppm.



This is the sources rule:
    repeat with second item running through gas sources:
        let space be the location of the second item;
        increase the former concentration of the space by
the contribution of the second item;
        if the former concentration of the space is less than
0.0ppm, now the former concentration of the space is
0.0ppm.



Test me with "z / z / z / z / z / z / z / z".
```

ExampleOnly You...

WI

Smoke which spreads through the rooms of the map, but only every other turn.

Suppose we want to have smoke that spreads from room to room, gradually filling the entire map with a clogging smoke. Having it spread every single turn would make for a pretty rapid diffusion, so we temper this by having it spread only on even-numbered turns, instead. Conveniently, Inform by default already knows about even and odd numbers, so we can write:

```
"Only You..."
```

Section 1 - The Procedure

Every turn when the turn count is even:
    if every room is smoky, make no decision;
    let previously smoky be whether or not the location is smoky;
    repeat with area running through smoky rooms:
        now every room which is adjacent to the area is smoky;
    if previously smoky is false and the location is smoky:
        say "[The location] is filling rapidly with smoke."

A room can be smoky or unsmoky.

Some air is a backdrop. Air is everywhere. Instead of doing something other than examining or smelling to air: say "It's just air." Understand "smoke" as the air when the location is smoky.

Instead of examining the air in a smoky room: say "A thick layer of smoke lies just under the ceiling."

Instead of smelling the air in a smoky room: say "Agh, acrid." Instead of smelling a smoky room: try smelling the air.

After looking in a smoky room: say "A thick layer of smoke has gathered under the ceiling."

Section 2 - The Scenario

The Guide Lodge is a room. "A very spacious room capable of containing several hundred girls while they eat, talk, or do crafts. It is constructed in a not-unappealing rustic style, with floor-to-ceiling windows overlooking the lake below, and a fieldstone hearth at the center." The Guide Lodge is smoky.

The Kitchen is north of the Guide Lodge. "Multiple eight-burner ranges, ovens, and a walk-in refrigerator: you know the sort of thing."

The Industrial Pantry is east of the Kitchen. "Awe-inspiring quantities of food line every shelf, from the three-gallon tub of mayonnaise to the 50-pound tub of rice. Perhaps the most astonishing item is a bag of marshmallows big enough to double as a futon."

The player is in the Pantry.

The Hallway is west of the Guide Lodge. The description of the Hallway is "A perpetually-crammed hallway which has to handle the overflow line for the toilets." A singed sign is fixed in place in the Hallway. The description of the sign is "Where the edge of the sign has not been burnt, the legible words are '...Can Prevent Forest Fires'."

The Toilets are north of the Hallway. "Always in full use, at least when the 12-to-15s are here."

The Coat Closet is south of the Hallway. "Muddy boots may not be worn inside the lodge; instead, about 250 pair are piled here, along with their owners' damp parkas and umbrellas."

The Craft Supply Room is west of the Hallway. "A holding-depot for jugs of white glue and popsicle sticks."

Test me with "x smoke / z / z / z / z / x smoke / look".

Example **Lethal Concentration 2**

Poisonous gas again, only this time it sinks.

This is a slight variation on the previous gas diffusion example: the main difference is that gas preferentially moves towards lower rooms, and will gradually settle in the bottom floor. We do this by calculating the probability of movement separately for each pair of rooms.

"Lethal Concentration"

A concentration is a kind of value. 200.9ppm specifies concentration. 200.9 ppm specifies concentration.

A room has a concentration called current concentration. A room has a concentration called former concentration.

To decide what number is the probability inverse between (space - a room) and (second space - a room):
    let guess be 20;
    let way be the best route from space to second space;
    if way is up, let guess be 50;
    if way is down, let guess be 10;
    if the guess is less than 10, decide on 10;
    decide on guess.

If we wanted, we could introduce other concerns into the calculation here: open and closed doors, windows between rooms, rooms that are outdoors vs. those that are indoors, and so on. The possibilities are numerous, so we will stick with the simple principle that our poison gas sinks.

```
Every turn:
    follow the diffusion rules.


The diffusion rules are a rulebook.


A diffusion rule (this is the gas movement rule):
    repeat with space running through rooms:
        let sum be 0.0 ppm;
        repeat with way running through directions:
            let second space be the room way from the
space;
            if second space is a room:
                let incoming be the former concentration of
the second space divided by the probability inverse
between second space and space;
                let outgoing be the former concentration of the
space divided by the probability inverse between space
and second space;
                let difference be incoming minus outgoing;
                increase sum by the difference;
        now current concentration of the space is the
former concentration of the space plus the sum.


A diffusion rule (this is the resetting concentration rule):
    repeat with space running through rooms:
        now the former concentration of the space is the
current concentration of the space.


The last diffusion rule (this is the lethal dosage rule):
    if the current concentration of the location is greater
than LC50:
        say "The concentration in the air overpowers
you...";
```

```
        end the story;
    otherwise:
        if the current concentration of the location is
greater than TLV-STEL:
            say "You feel extremely uncomfortable in this
environment."
```

```
Instead of doing something when the current
concentration of the location is greater than TLV-STEL:
    if going, continue the action;
    say "You can't work in this environment: your eyes
and nose sting and it hurts to breathe."
```

```
Room 1A is west of Room 1B. Room 1B is west of
Room 1C. Room 1C is west of Room 1D. Room 1D is
west of Room 1E.
```

```
Room 2A is west of Room 2B and below room 1A.
Room 2B is west of Room 2C and below Room 1B.
Room 2C is west of Room 2D and below Room 1C.
Room 2D is west of Room 2E and below Room 1D.
Room 2E is south of Room 1E and below Room 1E.
```

```
The former concentration of Room 1C is 800.0 ppm.
```

```
The status grid is a device carried by the player. The
status grid is switched on.
```

And just for fun, this time we'll make the grid prettier, too; but this will work only on the Z-machine setting, not Glulx.

```
Every turn:
    try examining the grid.
```

```
    Instead of examining the status grid:
    say "[unicode box drawings light down and right][top
bar][unicode box drawings light down and left][line
break]";
    say "[unicode box drawings light vertical]";
    say "[state of room 1A][state of room 1B][state of
room 1C][state of room 1D][state of room 1E]
upstairs[line break]";
    say "[unicode box drawings light vertical and right]
[middle bar][unicode box drawings light vertical and left]
[line break]";
    say "[unicode box drawings light vertical]";
    say "[state of room 2A][state of room 2B][state of
room 2C][state of room 2D][state of room 2E]
downstairs[line break]";
    say "[unicode box drawings light up and right][bottom
bar][unicode box drawings light up and left][variable
letter spacing][line break]"



Include Unicode Character Names by Graham Nelson.



To say top bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say
"[unicode box drawings light down and horizontal]";
        otherwise say "[unicode box drawings light
horizontal]".



To say middle bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say
"[unicode box drawings light vertical and horizontal]";
        otherwise say "[unicode box drawings light triple
dash horizontal]".



To say bottom bar:
    repeat with N running from 1 to 9:
        if the remainder after dividing N by 2 is 0, say
"[unicode box drawings light up and horizontal]";
        otherwise say "[unicode box drawings light
horizontal]".
```

TLV is a concentration that varies. TLV is 30.0ppm. [Long-term exposure maximum, safe for 8 hours a day.]

TLV-STEL is a concentration that varies. TLV-STEL is 50.0ppm. [Short-term exposure maximum, safe for fifteen minutes max.]

TLV-C is a concentration that varies. TLV-C is 150.0ppm. [Absolute exposure ceiling.]

LC50 is a concentration that varies. LC50 is 300.0ppm. [Concentration at which 50 percent of test subjects die of exposure, usually expressed in terms of time and body weight; in our LC50 these are factored in for the player's weight for one minute.]

Include Basic Screen Effects by Emily Short.

To say state of (space - a room):
    if the current concentration of space is less than TLV, say blue letters;
    if the current concentration of space is TLV, say blue letters;
    if the current concentration of space is greater than TLV, say green letters;
    if the current concentration of space is greater than TLV-STEL, say yellow letters;
    if the current concentration of space is greater than TLV-C, say red letters;
    say "[unicode square with diagonal crosshatch fill]";
    say default letters;
    say "[unicode box drawings light vertical]".

Test me with "z / z / z / z / z / z / z / z".

117 Example Thirst

A waterskin that is depleted as the player drinks from it.

"Thirst"

The player carries a waterskin. The waterskin can be full, partly drained, or empty. The waterskin is full. Understand "water" as the waterskin.

Instead of drinking the waterskin when the waterskin is empty:
    say "There is no water left."

Instead of drinking the waterskin: if the waterskin is partly drained, now the waterskin is empty; if the waterskin is full, now the waterskin is partly drained; say "You drink a long draught."

After printing the name of the waterskin: say " ([waterskin condition])"

Campsite is a room. "It is solid night now, and the stars have come out. Unfamiliar stars. On the other side of the valley -- a valley round-bottomed but shallow, like a soup bowl -- burn other campfires, most likely bandits. Their voices do not carry, but the smoke rises and obscures the starlight over that way."

A sleepsack is an enterable container in the Campsite.

"Your sleepsack is laid out in a pocket of sandy soil and coarse grass."

The sandy soil, the stars, the distant campfires, and the coarse grass are scenery in the Campsite. Understand "smoke" as the campfires. Instead of listening in the presence of your campfire: say "All you hear are the reassuring snaps and cracks of the sticks in your fire." Understand "campfires" or "fires" as the distant campfires.

Your campfire is scenery in the Campsite. Instead of pushing, pulling, turning, tasting, or touching your campfire, say "You would burn yourself." Understand "fire" as your campfire. The description of your campfire is "A reassuring protection against wild animals and cold."

The description of the stars is "You invent constellations for them. The slingshot. The scroll. The heart (upside down)."

Instead of going nowhere when the player is in Campsite:
    say "Now is not the time for wandering, alone in the dark. Better to keep here[if your campfire is visible], by the fire[end if]."

Singing is an action applying to nothing. Understand "sing" as singing.

Instead of singing:
    say "You sing, deep and low, a song from home. It is a good night for singing and the song raises your spirits."

Test me with "i / drink water / i / drink water / i".

---

122 **Example** **Beverage Service** *WI*

A potion that the player can drink.

Some kinds of game objects -- food, for instance -- can only sensibly be used once, and should then be destroyed. The EAT command already implements this, but suppose we also had a category of drinkable potions:

```
"Beverage Service"

A potion is a kind of thing. The sparkly blue potion is a potion carried by the player.

Level 3 is a room.

Instead of drinking a potion (called the drink):
    now the drink is nowhere;
    say "You quaff [the drink]. It goes down beautifully."

Test me with "drink sparkly / i".
```

---

403 **Example** **Flotation** *WI*

Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

Here we want a rulebook to determine whether objects float or sink, so we create an object-based rulebook for the purpose. The more specific rules here, pertaining to corks and to inflated things, will be consulted first; then, as a default, the general flotation rule.

We also want a switch that can turn flotation off at will. The rule about the big switch will be observed before the others because the when... clause makes it more specific than the other rules in the flotation rulebook.

If we wanted, we could also put these rules into a rulebook in an explicit order, overriding Inform's automatic sorting by specificity.

```
"Flotation"


The Pumping House is a room.


A well is a fixed in place container in the Pumping
House.


Instead of examining the well:
    say "[if something is in the well]On the surface of the
water you can see [a list of things in the well]
[otherwise]There is nothing on the surface of the water,
nor can you see into the depths[end if]."


The well bottom is a container.


The cork, the rubber ring and a lead ingot are in the
Pumping House.


A big switch is a fixed in place device in the Pumping
House. "A big switch labelled 'MAKE EVERYTHING
SINK' is mounted on one wall[if switched on]. It crackles
with electricity[otherwise]. It is currently switched off and
```

```
silent[end if]."
```

A thing can be inflated or uninflated. A thing is usually uninflated. Before printing the name of an inflated thing: say "inflated ".

The rubber ring is inflated.

The flotation rules are an object-based rulebook.

A flotation rule for the cork: rule succeeds.
A flotation rule for an inflated thing: rule succeeds.
A flotation rule when the big switch is switched on: rule fails.

```
After inserting something into the well:
    follow the flotation rules for the noun;
    if the rule succeeded:
        say "[The noun] bobs on the surface.";
    otherwise:
        move the noun to the well bottom;
        say "[The noun] sinks out of sight."
```

A thing can be sinking, rising, or static. A thing is usually static.

```
Definition: a thing is wet:
    if it is in the well, yes;
    if it is in the well bottom, yes;
    no.
```

Every turn:

```
    now every thing is static;
    repeat with item running through wet things:
        follow the flotation rules for the item;
        if the rule failed and the item is in the well, now the
item is sinking;
        if the rule succeeded and the item is in the well
bottom, now the item is rising;
    now every rising thing is in the well;
    now every sinking thing is in the well bottom;
    if something is rising, say "[The list of rising things]
rise[if the number of rising things is 1]s[end if] to the
surface of the well.";
    if something is sinking, say "[The list of sinking things]
sink[if the number of sinking things is 1]s[end if] out of
sight."
```

And finally a few description rules to make things look prettier:

```
Rule for writing a paragraph about the well when the
well contains something:
    say "The chief feature of the room is a concrete-sided
well in which there float[if the number of things in the
well is 1]s[end if] [a list of things in the well]."
```

```
Rule for writing a paragraph about the well:
    say "The chief feature of the room is a concrete-sided
well full of water."
```

As we recall from the chapter on activities, "writing a paragraph about..." is an activity; activities are themselves structured as sets of object-based rulebooks. The activity "writing a paragraph about" uses three object-based rulebooks (before writing..., for writing..., after writing...). We could have made a flotation activity as well, but in general it is overkill to make an activity to make success/failure decisions. For that purpose an object-based rulebook is sufficient.

```
Test me with "get all / put cork in well / put ring in well /
put ingot in well / x well / get cork / get ring / switch
switch on / put cork in well / put ring in well / x well /
switch switch off / switch switch on".
```

Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

If we wanted to define a brand new verb that did affect a specific object, we might begin like this:

```
"Xylan"

Understand "hydrolyze [something]" as hydrolyzing.
Hydrolyzing is an action applying to one thing.

Carry out hydrolyzing:
    say "[The noun] cannot be hydrolyzed."

Instead of hydrolyzing the xylan:
    move the xylose to the holder of the xylan;
    now the xylan is nowhere;
    say "At once the xylan becomes xylose."

Plant Cell Wall is a room.

There is a xylose sample. The xylan sample is a thing in
Plant Cell Wall. The description of the xylan is "A
polysaccharide. Totally useless. If only you had some
xylose, instead!" The description of the xylose is
"Awesome!"

Test me with "x xylan / hydrolyze xylan / x xylose".
```

Of course, how our players will ever solve this problem is another

question (especially if their biology and chemistry are both rusty). When adding entirely new commands to a game, it is often a good idea to provide as many ways of phrasing the command as possible; to drop hints about the correct phrasing within the game's text; or even to tell the player about the expanded command list in some documentation or help at the beginning of the game. So for instance we might also add

```
Understand "break down [something] with water" or
"break [something] down with water" as hydrolyzing.
```

And these lines will also provide syntax for our new command, without interfering with the previous syntax. It's also good to anticipate alternative (British or American) spellings. People's typing habits are hard to overcome, even if they know you are spelling the word the other way. It is probably best not to annoy them unduly. So:

```
Understand "hydrolyse [something]" as hydrolyzing.
```

Then some text in-game might offer a clue, subtle or (since this is an example) blunt:

```
Instead of examining the player, say "You're a drop of
water, which means that you can break down certain
chemicals!"
```

```
Understand "break down [something]" or "break
[something] down" as hydrolyzing.
```

And finally, we could try adding instructions explicitly:

```
Understand "help" or "hint" or "hints" or "instructions" or
"info" or "about" as asking for help. Asking for help is an
action out of world. Carry out asking for help: say "The
following commands are understood, in addition to the
standard ones: EVAPORATE, FREEZE, HYDROLYZE,
SUBLIME..."
```

Test more with "help / x me / break down xylan"

...though of course in fact these other commands won't be available until we define them, too.

This last approach, defining all the extra commands up front, is especially useful if these commands are very technical or unusual; if they are needed early in the game, before you've a chance to educate the player; or if they are not suggested by any in-game objects. A player who encounters a tool with an obvious use, such as a hairbrush, will likely think of trying to BRUSH things with it. It's harder to rely on his guessing actions that are both outside the range of usual commands and unrelated to any of the visible props, however.

---

192 ExampleFrizz *WI*

Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

Suppose we have some items that get wet in contact with other damp things; in particular, if we touch anything wet while wearing a pair of gloves, the gloves too get damp. This requires that we be systematic about detecting all cases where contact occurs. So:

"Frizz"

A thing can be waterproof or porous. A container is usually waterproof. An animal is usually waterproof.

A thing can be sodden or dry.

A container can be waterfilled or empty.

Before printing the name of a sodden thing: say "sodden

". Understand the sodden property as describing a thing.

The player wears a pair of woolly gloves. Instead of wearing a sodden thing: say "You dubiously contemplate [the noun], and decide it's best to wear dry clothing."

Every turn:
    follow the liquid distribution rules.

The liquid distribution rules is a rulebook.

A liquid distribution rule:
    repeat with item running through containers:
        if the item is open and the item is empty and the item is in a waterfilled container:
            now the item is waterfilled;
            if the player can see the item, say "[The item] fills up with water, of course.".

A liquid distribution rule:
    repeat with item running through things in a waterfilled container:
        if the item is porous and the item is dry:
            if the player can see the item, say "[The item] soaks through.";
            now the item is sodden.

A liquid distribution rule:
    repeat with item running through sodden things in the airing cupboard:
        if the item is not The Last Man, now the item is dry.

The last liquid distribution rule:
    if the player carries a dry copy of The Last Man, end the story finally.

Instead of examining a waterfilled container:
    say "[The noun] is full of water[if the noun contains something visible]; it also contains [the list of things in the noun][end if]."

Instead of examining a container:
    say "Dry inside[if the noun contains something visible], and containing [a list of things in the noun]."

A book is a kind of thing. Instead of examining a sodden book, say "[The noun] is too soaked to read, thanks to someone's carelessness." Understand "book" as a book.

The copy of The Last Man is a book. The description of The Last Man is "Mary Shelley's very own original copy, loaned to you under the strictest of agreements[if the pond encloses the copy]. How it came to be in its present position is a long story, and not important at the moment: the critical thing is not to blame oneself - who could have anticipated the cricket bat, anyway? - but to fix it immediately, before anything worse occurs[end if]." The The Last Man is in the ziploc bag. The ziploc bag is waterproof and empty. It is openable, transparent, and closed.

The Back Garden is a room. The Back Garden contains a pond. The pond is a waterfilled container. In the pond is the ziploc bag. A carp is in the pond. The carp is an animal. Instead of taking the carp, say "You're not fast enough."

Rule for writing a paragraph about the pond:
    if the pond encloses the Last Man:
        say "It is a beautiful day just at the end of spring and beginning of summer. The sun shines, the trees blossom, the world conspires in Edenic cheerfulness. You can take in none of it. Your eye is on [the Last Man].";
    otherwise:
        say "The pond[if something is in the pond] (containing [the list of things in the pond])[end if] remains a figure of menace in an otherwise lushly verdant landscape."

Before printing the name of the Last Man when the Last Man is enclosed by the Pond: if the Last Man is dry, say "delicate, valuable ".

After printing the name of something (called the target) while writing a paragraph about something:
    if the Last Man is enclosed by the Pond:
        if the target is in something (called the parent), say " - which is in [the parent]".

The Kitchen is west of the Back Garden. "Not in any sense your province: Mrs Peaswell gets agitated if anyone besides herself so much as boils a kettle. In general it is best to sneak through, disturbing as little as possible and preserving the cosy domestic fiction that academics cannot cook.

The nearby stairwell leads up to the second floor."

The Stairwell is above the Kitchen. "Halfway up and halfway down." The airing cupboard is a container in the Stairwell. "An airing cupboard here contains the heating system, and is the ideal place for restoring wet items to a dry state." The airing cupboard is fixed in place.

Now: whenever the player definitely touches something, we want to

follow certain rules about the transfer of liquid. These rules need to come after an action's check rules (to make sure the action really occurs) and before the carry out rules (so that nothing has moved or changed yet). So we'll borrow from the chapter on rulebooks to create a whole new stage to the action, occurring between the check and the carry out phases:

```
The post-check rules are a rulebook.


This is the post-check stage rule:
    abide by the post-check rules.


The post-check stage rule is listed before the carry out
stage rule in the specific action-processing rules.


A post-check rule (this is the dry glove rule):
    if we get wet:
        if the player wears the gloves and the gloves are
dry:
            now the gloves are sodden;
            say "(soaking your gloves in the process)";
    continue the action.


A post-check rule (this is the wet glove rule):
    if the player wears the sodden gloves:
        if the The Last Man must be touched and the Last
Man is not sodden:
            say "(soaking the parched pages of The Last
Man with the rude touch of your sodden gloves)";
            now the The Last Man is sodden;
        continue the action.


Before doing something when the player does not wear
the gloves:
    if The Last Man must be touched:
        if The Last Man is dry, say "[The The Last Man] is
too precious to endanger when you are not wearing
```

gloves." instead;
        otherwise say "You hesitate instinctively, then recollect that you can hardly harm [The The Last Man] any more than it has already been harmed...".

To decide whether we get wet:
    if the noun is not a thing, no;
    if the noun dampens us, yes;
    if the second noun is not a thing, no;
    if the second noun dampens us, yes;
    no.

To decide whether (item - a thing) dampens us:
    if the item is not liquiferous, no;
    if item must be touched, yes;
    no.

To decide whether (item - a thing) must be touched:
    if the item is the noun and the action requires a touchable noun, yes;
    if the item is the second noun and the action requires a touchable second noun, yes;
    no.

Definition: a thing is liquiferous:
    if it is sodden, yes;
    if it is in a waterfilled container, yes;
    no.

Test me with "x book / x bag / get bag / take off gloves / w / u / put gloves in cupboard / z / open bag / touch book / push book / turn book / get gloves / wear gloves / get book".

Test disaster with "open bag".

Test mishandling with "get bag / open bag / get book".

Example 3 AM

A shake command which agitates soda and makes items thump around in boxes.

"3 AM"

Understand "shake [something preferably held]" as shaking.

Shaking is an action applying to one carried thing.

Carry out shaking:
    say "Nothing results of your shaking [the noun]."

Instead of shaking a closed container when something is in the noun:
    say "Something rattles inside [the noun]."

Instead of shaking a closed transparent container when something is in the noun:
    say "Inside [the noun] there are banging noises produced by [the list of things contained by the noun]."

Instead of shaking an open container which contains something:

```
    say "[The list of things contained by the noun] might
fly out."
```

```
The Wawa is a room. "A convenience store, if you like
to call it that, vending the usual assortment of chips,
donuts, soda, and beer. There is something of a line at
the sandwich counter."
```

```
The box of enrobed cakes is in the Wawa. "A box of
Tastykake Enrobed Cakes has fallen off its shelf." The
description is "'Enrobed Cakes' is a fancy term for
'strange sponge-like baked good, covered in a thin shell
of waxy chocolate'. They are addictive, but not in a way
that lets you respect yourself in the morning." The box is
a closed openable container. In the box is a cake.
```

```
Instead of opening the box, say "The Wawa clerks frown
on the consumption of unpurchased foodstuffs."
```

```
The can of root beer is a closed openable container
carried by the player. The can of root beer is either
agitated or calm.
```

Because the can of root beer should have some reactions to having been shaken later in the game, we need to borrow a few ideas from the chapter on Time:

```
Instead of shaking the can of root beer:
    the can calms down in five turns from now;
    say "You give the can a good hard shake.";
    now the can is agitated.
```

```
Instead of listening to the can: say "It sounds [if
agitated]fizzy[otherwise]calm[end if]!"
```

```
At the time when the can calms down:
    now the can is calm.


The sticky mess is fixed in place. "There is a sticky
mess on the ground."


Instead of opening the agitated can of root beer:
    now the can of root beer is nowhere;
    now the sticky mess is in the location;
    say "You open the can and fizzing sweet soda goes
absolutely everywhere."


Instead of opening the calm can of root beer when the
can has been agitated:
    now the can of root beer is nowhere;
    say "The root beer is disappointingly flat. That's what
you get for shaking it up!"


Test me with "get box / shake box / open box / shake
box / listen to can / shake can / listen to can / wait / wait
/ wait / wait / wait / listen to can / open can".
```

ExampleLemonade

Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

Liquids, and all substances that can be mixed or broken off in partial amounts, pose a challenge to model in interactive fiction. The following example is a simple one, but adequate for many scenarios.

We start by assuming that all liquids in the game will always appear in containers. The player can pour liquids from one container to another, and the containers keep track of how full they are and describe themselves appropriately. The player can also refer to containers by content.

Mixture, however, is not allowed, nor is it possible to put liquids on other objects, pour them out on the ground, etc. These ideas would require a more complicated set-up.

```
"Lemonade"

A volume is a kind of value. 15.9 fl oz specifies a
volume with parts ounces and tenths (optional,
preamble optional).

A fluid container is a kind of container. A fluid container
has a volume called a fluid capacity. A fluid container
has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl
oz. The current volume of a fluid container is usually 0.0
fl oz.

Liquid is a kind of value. The liquids are water, milk,
lemonade, and iced tea. A fluid container has a liquid.

Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun]
at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [liquid of the noun]."

To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3
and tenths part of amount is less than 7:
```

```
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in
words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase
amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid
ounce[s]".



Before printing the name of a fluid container (called the
target) while not drinking:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.



After printing the name of a fluid container (called the
target) while not examining:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.



Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."



Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.



Understand "drink from [fluid container]" as drinking.



Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within."
instead;
```

```
    otherwise:
        decrease the current volume of the noun by 0.2 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "You take a sip of [the liquid of the noun][if the
noun is empty], leaving [the noun] empty[end if]."
```

We have allowed all liquids to be drunk, but it would be possible also to add checking, if we had a game where some liquids were beverages and others were, say, motor oil.

```
Understand the command "fill" as something new.


Understand "pour [fluid container] in/into/on/onto [fluid
container]" as pouring it into. Understand "empty [fluid
container] into [fluid container]" as pouring it into.
Understand "fill [fluid container] with/from [fluid
container]" as pouring it into (with nouns reversed).


Understand "pour [something] in/into/on/onto
[something]" as pouring it into. Understand "empty
[something] into [something]" as pouring it into.
Understand "fill [something] with/from [something]" as
pouring it into (with nouns reversed).


Pouring it into is an action applying to two things.


Check pouring it into:
    if the noun is not a fluid container, say "You can't pour
[the noun]." instead;
    if the second noun is not a fluid container, say "You
can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly
pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second
noun:
        if the second noun is empty, now the liquid of the
second noun is the liquid of the noun;
```

```
        otherwise say "Mixing [the liquid of the noun] with
    [the liquid of the second noun] would give unsavory
    results." instead;
        if the noun is empty, say "No more [liquid of the noun]
    remains in [the noun]." instead;
        if the second noun is full, say "[The second noun]
    cannot contain any more than it already holds." instead.


    Carry out pouring it into:
        let available capacity be the fluid capacity of the
    second noun minus the current volume of the second
    noun;
        if the available capacity is greater than the current
    volume of the noun, now the available capacity is the
    current volume of the noun;
        increase the current volume of the second noun by
    available capacity;
        decrease the current volume of the noun by available
    capacity.


    Report pouring it into:
        say "[if the noun is empty][The noun] is now empty;
    [otherwise][The noun] now contains [current volume of
    the noun in rough terms] of [liquid of the noun]; [end if]";
        say "[the second noun] contains [current volume of
    the second noun in rough terms] of [liquid of the second
    noun][if the second noun is full], and is now full[end if]."
```

This is probably a drier description than we would actually want in our story, but it does allow us to see that the mechanics of the system are working, so we'll stick with this for the example.

Now we need a trick from a later chapter, which allows something to be described in terms of a property it has. This way, the story will understand not only "pitcher" and "glass" but also "pitcher of lemonade" and "glass of milk" -- and, indeed, "glass of lemonade", if we empty the glass and refill it with another substance:

```
    Understand the liquid property as describing a fluid
    container. Understand "of" as a fluid container.
```

And now the scenario itself:

The Porch is a room. The porch swing is an enterable supporter in the Porch. "An inviting swing hangs here at the end of the porch, allowing you to enjoy the summer with a cool beverage, and watch your neighbor Ted mowing his lawn with the very last manual powerless lawnmower on the block."

The glass is a fluid container carried by the player. The liquid of the glass is milk. The current volume of the glass is 0.8 fl oz.

The pitcher is a fluid container in the Porch. The fluid capacity of the pitcher is 32.0 fl oz. The current volume of the pitcher is 20.0 fl oz. The liquid of the pitcher is lemonade.

Ted's Lawn is outside from the Porch. Ted is a man in Ted's Lawn. "Ted has taken off his shirt, but still seems a bit oppressed by the sun." The description of Ted is "He looks hot. In all senses."

After deciding the scope of the player: place Ted in scope.

Instead of doing something to Ted when the player is in the Porch: say "You can't really interact with Ted from this distance, except in the sense of eyeing him surreptitiously."

Instead of giving an empty fluid container to Ted: say "Yes, taunt the poor man, why don't you?"

```
Instead of giving a fluid container to Ted when the liquid
of the noun is milk: say "Ted looks ruefully at the milk.
'Thanks, but I'm lactose-intolerant,' he says."


The block giving rule is not listed in the check giving it to
rules.


Every turn:
    if Ted is in the location:
        if Ted carries a fluid container (called refreshment):
            try Ted drinking the refreshment;
        otherwise if a random chance of 1 in 3 succeeds:
            say "Ted pushes the ineffective mower over
some dandelions."


Instead of someone drinking a fluid container:
    if the noun is empty:
        try the person asked giving the noun to the player;
    otherwise:
        decrease the current volume of the noun by 2.0 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "[The person asked] gulps down some [liquid
of the noun]."


After someone giving something to the player:
    say "'Here,' says [the person asked], handing [the
noun] back to you. 'Thanks, I owe you one.'";
    end the story finally.


Test me with "x milk / x lemonade / drink lemonade /
drink milk / pour lemonade into glass / drink milk / x milk
/ drink milk / g / i / fill glass with lemonade / drink
lemonade / drop glass / drink lemonade / pitcher".
```

Test Ted with "out / give milk to ted / drink milk / g / g / g / give glass to ted / in / fill glass with lemonade / out / give lemonade to ted / wait / z / z / z ".

Example **Savannah** *WI*

267

Using the liquid implementation demonstrated in Lemonade for putting out fires.

Here we build very slightly on the existing liquid implementation to add a puzzle where the player puts out a fire with a bucket of water. Most of the liquid implementation remains the same as before, but now we understand the names of containers according to the liquids they contain.

The new material, pertaining to extinguishing fires, is at the bottom in section 2.

"Savannah"

Section 1 - Essentials of Liquid

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The current volume of a fluid container is usually 0.0 fl oz.

```
Liquid is a kind of value. A fluid container has a liquid.


Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun]
at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [liquid of the noun]."


To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3
and tenths part of amount is less than 7:
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in
words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase
amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid
ounce[s]".


Before printing the name of a fluid container (called the
target) while not drinking:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.


After printing the name of a fluid container (called the
target) while not examining:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.


Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
```

accept anything but liquids."

Definition: a fluid container is empty if the current volume of it is 0.0 fl oz. Definition: a fluid container is full if the current volume of it is the fluid capacity of it.

Understand "drink from [fluid container]" as drinking.

Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within." instead;
    otherwise:
        decrease the current volume of the noun by 0.2 fl oz;
        if the current volume of the noun is less than 0.0 fl oz, now the current volume of the noun is 0.0 fl oz;
        say "You take a sip of [the liquid of the noun][if the noun is empty], leaving [the noun] empty[end if]."

Understand the command "fill" as something new.

Understand "pour [fluid container] in/into/on/onto [fluid container]" as pouring it into. Understand "empty [fluid container] into [fluid container]" as pouring it into. Understand "fill [fluid container] with/from [fluid container]" as pouring it into (with nouns reversed).

Understand "pour [something] in/into/on/onto [something]" as pouring it into. Understand "empty [something] into [something]" as pouring it into. Understand "fill [something] with/from [something]" as pouring it into (with nouns reversed).

Pouring it into is an action applying to two things.

Check pouring it into:
    if the noun is not a fluid container, say "You can't pour [the noun]." instead;
    if the second noun is not a fluid container, say "You can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second noun:
        if the second noun is empty, now the liquid of the second noun is the liquid of the noun;
        otherwise say "Mixing [the liquid of the noun] with [the liquid of the second noun] would give unsavory results." instead;
    if the noun is empty, say "No more [liquid of the noun] remains in [the noun]." instead;
    if the second noun is full, say "[The second noun] cannot contain any more than it already holds." instead.

Carry out pouring it into:
    let available capacity be the fluid capacity of the second noun minus the current volume of the second noun;
    if the available capacity is greater than the current volume of the noun, now the available capacity is the current volume of the noun;
    increase the current volume of the second noun by available capacity;
    decrease the current volume of the noun by available capacity.

Report pouring it into:
    say "[if the noun is empty][The noun] is now empty; [otherwise][The noun] now contains [current volume of the noun in rough terms] of [liquid of the noun]; [end if]";
    say "[the second noun] contains [current volume of the second noun in rough terms] of [liquid of the second noun][if the second noun is full], and is now full[end if]."

```
Understand the liquid property as describing a fluid
container. Understand "of" as a fluid container.
```

```
Section 2 - Putting Out Fires
```

```
The Beach is a room. "The Atlantic stretches east to the
horizon, though it is at low tide at the moment. It is
dawn: time to pack up and go home."
```

We will skip implementing the Pacific ocean itself, though the example Lakeside Living shows how to incorporate large bodies of water into our liquid simulation.

```
The liquids are seawater. [We could include others, but
for the moment...]
```

```
Instead of drinking a fluid container when the liquid of
the noun is seawater:
    say "Blech!"
```

```
The bucket is a fluid container carried by the player. The
liquid of the bucket is seawater. The current volume of
the bucket is 64.0 fl oz.
```

```
The fire is a fixed in place thing in the beach. "A low fire
crackles here, left over from an attempt at s'mores much
earlier in the evening."
```

```
Instead of touching or rubbing or taking the fire, say
"You're not such a glutton for punishment."
```

```
Instead of pouring something into the fire:
    now the fire is nowhere;
    now the current volume of the noun is 0.0 fl oz;
    say "[The second noun] goes out in a great hiss."


Test me with "drink seawater / pour seawater on fire / x
bucket / i".
```

This is still a specific implementation: if we wanted to weave liquids together with a full-scale burning model (as in "In Fire or in Flood"), where pretty much any object in the game can be flaming (currently on fire) or damp (extinguished), we might generalize our rule to

```
Instead of pouring something into a flaming thing:
    now the second noun is damp;
    now the current volume of the noun is 0.0 fl oz;
    say "[The second noun] goes out in a great hiss."
```

Of course, the merging of fire and liquids also raises the possibility of gasoline and explosives, of heating and boiling liquids, etc.: as always, it's wise to incorporate a simulation that is only as detailed as the game's interactions really justify.

ExampleNoisy Cricket *WI*

Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

Our previous experiments into liquid have not dealt with the possibility of mixing components, but that is because for most games, tracking the details of mixture is overkill.

But let's suppose that this time we do want to have mixed liquids; moreover, we want a way to describe the mixtures to the player inventively, so that if he hits specific combinations those combinations are recognized: calling the result a martini, say, rather than just "a mixture of vodka and vermouth".

The implementation that follows relies on a fairly simple idea from linear algebra. Any given liquid can be expressed as a vector in N-space, where N is the number of available ingredients and the length of the

vector depends on how much of each ingredient is used; then we find the recipe that best describes the liquid by taking the dot product of our liquid vector with a bunch of sample vectors and selecting the one with the largest result.

If this does not make sense, don't worry: it's not necessary to understand the idea to use the code.

Any implementation involving a large number of place values is always a bit challenging in integer arithmetic. This examples assumes that no bodies of liquid will ever be very large, and that the proportions of ingredients in a mixture will not be vastly askew. (No 20-parts-to-1 proportions, for instance.) This probably works reasonably well for the cocktails that we make the basis of the example.

"Noisy Cricket"

Part 1 - Volumes and Mixtures

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called creme de menthe volume. A fluid container has a volume called vodka volume. A fluid container has a volume called cacao volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The creme de menthe volume of a fluid container is usually 0.0 fl oz. The vodka volume of a fluid container is usually 0.0 fl oz. The cacao volume of fluid container is usually 0.0 fl oz.

To decide what volume is the current volume of (item - a

```
fluid container):
    let total be the creme de menthe volume of the item;
    increase total by the vodka volume of the item;
    increase total by the cacao volume of the item;
    decide on total.


Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the nominal descriptor
of the noun] at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [adjectival descriptor of the
noun] [nominal descriptor of the noun]."


Adjectival descriptor is a kind of value. The adjectival
descriptors are strong, chocolatey, minty, perfect, and
pure.


Nominal descriptor is a kind of value. The Nominal
descriptors are creme de menthe, vodka, creme de
cacao, grasshopper, chocolate vodka, mint vodka,
chocolate martini, mintini, chocolate mint martini.
```

Our table of mixtures is expressed in parts: so if a recipe contains one part X and two parts Y, we would put "1" in the first column and "2" in the second column.

```
Table of Mixtures

[illegible]comp
[illegible]de menthe
[illegible]
[illegible]tey
[illegible]cacao
[illegible]tey
[illegible]
[illegible]tey
[illegible]per
[illegible]tey martini
[illegible]
[illegible]tey vodka
[illegible]
[illegible]tey mint martini
```

A fluid container has an adjectival descriptor. A fluid container has a nominal descriptor. Understand the adjectival descriptor property as describing a fluid container. Understand the nominal descriptor property as describing a fluid container.

To decide what number is (quantity - a number) squared:
    decide on quantity times quantity.

To score mixtures in (item - a fluid container):
    repeat through Table of Mixtures:
        let total line parts be creme de menthe comp entry squared;
        let total line parts be total line parts plus vodka comp entry squared;
        let total line parts be total line parts plus cacao comp entry squared;
        let creme de menthe score be creme de menthe comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the vodka volume of item;
        let cacao score be cacao comp entry times the cacao volume of item;
        let total score be creme de menthe score plus vodka score;
        let total score be total score plus cacao score;
        let total score be total score times calibration for total line parts;
        now rating entry is total score;
        if total line parts is 1, now adjectival descriptor entry is pure;
        otherwise now adjectival descriptor entry is perfect;
        [and for creme de menthe...]
        now creme de menthe comp entry is creme de menthe comp entry plus 1;
        let total line parts be creme de menthe comp entry squared plus vodka comp entry squared;
        let total line parts be total line parts plus cacao comp entry squared;
        let creme de menthe score be creme de menthe comp entry times the creme de menthe volume of item;
        let vodka score be vodka comp entry times the vodka volume of item;

```
	let cacao score be cacao comp entry times the
cacao volume of item;
	let total score be creme de menthe score plus
vodka score;
	let total score be total score plus cacao score;
	let total score be total score times calibration for
total line parts;
	if total score is greater than rating entry, now
adjectival descriptor entry is minty;
	now creme de menthe comp entry is creme de
menthe comp entry minus 1;
	[and for vodka...]
	now vodka comp entry is vodka comp entry plus 1;
	let total line parts be creme de menthe comp entry
squared plus vodka comp entry squared;
	let total line parts be total line parts plus cacao
comp entry squared;
	let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
	let vodka score be vodka comp entry times the
vodka volume of item;
	let cacao score be cacao comp entry times the
cacao volume of item;
	let total score be creme de menthe score plus
vodka score;
	let total score be total score plus cacao score;
	let total score be total score times calibration for
total line parts;
	if total score is greater than rating entry, now
adjectival descriptor entry is strong;
	now vodka comp entry is vodka comp entry minus
1;
	[and for cacao...]
	now cacao comp entry is cacao comp entry plus 1;
	let total line parts be creme de menthe comp entry
squared plus vodka comp entry squared;
	let total line parts be total line parts plus cacao
comp entry squared;
	let creme de menthe score be creme de menthe
comp entry times the creme de menthe volume of item;
	let vodka score be vodka comp entry times the
vodka volume of item;
	let cacao score be cacao comp entry times the
cacao volume of item;
	let total score be creme de menthe score plus
vodka score;
	let total score be total score plus cacao score;
	let total score be total score times calibration for
total line parts;
	if total score is greater than rating entry, now
adjectival descriptor entry is chocolatey;
```

```
        now cacao comp entry is cacao comp entry minus
1.



To identify mixture in (item - a fluid container):
    score mixtures in item;
    sort Table of Mixtures in reverse rating order;
    choose row 1 in Table of Mixtures;
    now nominal descriptor of the item is nominal
descriptor entry;
    let sample vodka be vodka comp entry; [Now keep
track of all these]
    let sample creme de menthe be creme de menthe
comp entry;
    let sample cacao be cacao comp entry;
    if rating entry divided by 100 is the current volume of
the item:
        now adjectival descriptor of the item is pure;
    otherwise:
        now adjectival descriptor of the item is adjectival
descriptor entry.



To decide what number is the raw quantity of (item
volume - a volume):
    let raw be item volume divided by 0.5 fl oz;
    decide on raw.



To decide what number is calibration for (total - a
number):
    if total is an initial listed in the table of Multipliers,
decide on result entry;
    decide on 21.
```

Here we cheat on our arithmetic. The following chart just provides values corresponding roughly to 1/(sqrt (x)), but since Inform does not deal very gracefully with square roots or fractions, we will calculate this elsewhere and just supply the answers in the code:

```
Table of Multipliers

initial  result
1        100
```

```
        do nothing.


After printing the name of a fluid container (called the
target) while not examining or pouring:
    unless the target is empty:
        say " of [adjectival descriptor of the target] [nominal
descriptor of the target]";
        omit contents in listing.


Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."


Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.


Understand "drink from [fluid container]" as drinking.


Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [nominal descriptor of the
noun] within." instead;
    otherwise:
        let cacao loss be the consumed cacao of the noun
out of sip volume;
        let creme de menthe loss be the consumed creme
de menthe of the noun out of sip volume;
        let vodka loss be the consumed vodka of the noun
out of sip volume;
        decrease the cacao volume of the noun by the
cacao loss;
        decrease the creme de menthe volume of the noun
by creme de menthe loss;
        decrease the vodka volume of the noun by vodka
loss;
        say "You take a sip of [the nominal descriptor of
the noun][if the noun is empty], leaving [the noun]
empty[end if].".
```

Sip volume is a volume that varies. Sip volume is 0.5 fl oz.

To decide what volume is the consumed cacao of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the cacao volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;
    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the cacao volume of the item, decide on the cacao volume of the item;
    decide on consumed volume.

To decide what volume is the consumed creme de menthe of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the creme de menthe volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;
    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the creme de menthe volume of the item, decide on the creme de menthe volume of the item;
    decide on consumed volume.

To decide what volume is the consumed vodka of (item - a fluid container) out of (total consumption - a volume):
    let new volume be the vodka volume of the item times 100;
    let percentage be the new volume divided by the current volume of the item;
    let consumed volume be the percentage times total consumption;

    let consumed volume be consumed volume divided by 100;
    if consumed volume is greater than the vodka volume of the item, decide on the vodka volume of the item;
    decide on consumed volume.

Part 2 - Filling

Understand the command "fill" as something new.

Understand "fill [something] with/from [something]" as filling it with.

Filling it with is an action applying to two things. Carry out filling it with: try pouring the second noun into the noun instead.

Understand "pour [fluid container] in/into/on/onto [fluid container]" as pouring it into. Understand "empty [fluid container] into [fluid container]" as pouring it into.

Understand "pour [something] in/into/on/onto [something]" as pouring it into. Understand "empty [something] into [something]" as pouring it into.

Pouring it into is an action applying to two things.

Check pouring it into:
    if the noun is not a fluid container, say "You can't pour [the noun]." instead;
    if the second noun is not a fluid container, say "You can't pour liquids into [the second noun]." instead;

if the noun is the second noun, say "You can hardly pour [the noun] into itself." instead;
if the noun is empty, say "No more [nominal descriptor of the noun] remains in [the noun]." instead;
if the second noun is full, say "[The second noun] cannot contain any more than it already holds." instead.

Carry out pouring it into:
let available capacity be the fluid capacity of the second noun minus the current volume of the second noun;
if the available capacity is greater than the current volume of the noun, now the available capacity is the current volume of the noun;
let cacao loss be the consumed cacao of the noun out of available capacity;
let creme de menthe loss be the consumed creme de menthe of the noun out of available capacity;
let vodka loss be the consumed vodka of the noun out of available capacity;
decrease the cacao volume of the noun by the cacao loss;
decrease the creme de menthe volume of the noun by creme de menthe loss;
decrease the vodka volume of the noun by vodka loss;
increase the cacao volume of the second noun by the cacao loss;
increase the creme de menthe volume of the second noun by creme de menthe loss;
increase the vodka volume of the second noun by vodka loss.

Report pouring it into:
identify mixture in noun;
identify mixture in second noun;
say "[if the noun is empty][The noun] is now empty; [otherwise][The noun] now contains [current volume of the noun in rough terms] of [nominal descriptor of the noun]; [end if]";
say "[the second noun] contains [current volume of the second noun in rough terms] of [adjectival descriptor of the second noun] [nominal descriptor of the second noun][if the second noun is full], and is now full[end if]."

Understand "of" as a fluid container.

Part 3 - Scenario

When play begins: say "When you decided to try Mixology WS102 (*cross-listed with Women's Studies), you envisioned yourself writing essays about gender discrimination during the Prohibition, say, or reading essays on male vs. female metabolism of alcohol. But no: MxWS102 turns out to be about... mixing the perfect chocolate mint martini."

The College of Mixology is a room. The bar is a supporter in the college.

The cocktail glass is a fluid container carried by the player. The fluid capacity of the cocktail glass is 4.0 fl oz.

The flask is a fluid container carried by the player. The vodka volume of the flask is 4.0 fl oz.

The jigger is a fluid container carried by the player. The fluid capacity of the jigger is 1.0 fl oz.

The small measure is a fluid container carried by the player. The fluid capacity of the small measure is 0.5 fl oz.

The decanter is a fluid container on the bar. The fluid capacity of the decanter is 32.0 fl oz. The creme de menthe volume of the decanter is 20.0 fl oz.

The bottle is a fluid container carried by the player. The cacao volume of the bottle is 10.0 fl oz.

Test me with "i / pour flask in jigger / pour jigger in glass / pour bottle in jigger / pour jigger in glass / pour bottle in jigger / pour jigger in glass / pour decanter in jigger / pour jigger in glass / drink glass / g / g / x glass / pour flask in glass".

---

333 **Example** **Lakeside Living** *WI*

Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

Much of what follows is identical to "Lemonade" earlier; the new material begins at Part 2.

"Lakeside Living"

A volume is a kind of value. 15.9 fl oz specifies a volume with parts ounces and tenths (optional, preamble optional).

A fluid container is a kind of container. A fluid container has a volume called a fluid capacity. A fluid container has a volume called current volume.

The fluid capacity of a fluid container is usually 12.0 fl oz. The current volume of a fluid container is usually 0.0 fl oz.

```
Liquid is a kind of value. The liquids are water, absinthe,
and iced tea. A fluid container has a liquid.


Instead of examining a fluid container:
    if the noun is empty,
        say "You catch just a hint of [the liquid of the noun]
at the bottom.";
    otherwise
        say "[The noun] contains [current volume of the
noun in rough terms] of [liquid of the noun]."


To say (amount - a volume) in rough terms:
    if the amount is less than 0.5 fl oz:
        say "a swallow or two";
    otherwise if tenths part of amount is greater than 3
and tenths part of amount is less than 7:
        let estimate be ounces part of amount;
        say "[estimate in words] or [estimate plus 1 in
words] fluid ounces";
    otherwise:
        if tenths part of amount is greater than 6, increase
amount by 1.0 fl oz;
        say "about [ounces part of amount in words] fluid
ounce[s]".


Before printing the name of a fluid container (called the
target) while not drinking or pouring:
    if the target is empty:
        say "empty ";
    otherwise:
        do nothing.


After printing the name of a fluid container (called the
target) while not examining or pouring:
    unless the target is empty:
        say " of [liquid of the target]";
        omit contents in listing.
```

```
Instead of inserting something into a fluid container:
    say "[The second noun] has too narrow a mouth to
accept anything but liquids."



Definition: a fluid container is empty if the current
volume of it is 0.0 fl oz. Definition: a fluid container is full
if the current volume of it is the fluid capacity of it.



Understand "drink from [fluid container]" as drinking.



Instead of drinking a fluid container:
    if the noun is empty:
        say "There is no more [liquid of the noun] within."
instead;
    otherwise:
        decrease the current volume of the noun by 0.2 fl
oz;
        if the current volume of the noun is less than 0.0 fl
oz, now the current volume of the noun is 0.0 fl oz;
        say "You take a sip of [the liquid of the noun][if the
noun is empty], leaving [the noun] empty[end if]."



Part 2 - Filling



Understand the command "fill" as something new.
```

Here we want Inform to prefer full liquid sources to other containers when it chooses an end to a player's unfinished or ambiguous command. And so:

```
Understand "fill [fluid container] with/from [full liquid
source]" as filling it with. Understand "fill [fluid container]
with/from [fluid container]" as filling it with.
```

Both grammar lines point to the same ultimate outcome; the purpose of specifying both is to tell Inform to check thoroughly for full liquid sources

before falling back on other fluid containers when making its decisions.

```
Understand "fill [something] with/from [something]" as
filling it with.


Filling it with is an action applying to two things. Carry
out filling it with: try pouring the second noun into the
noun instead.


Understand "pour [fluid container] in/into/on/onto [fluid
container]" as pouring it into. Understand "empty [fluid
container] into [fluid container]" as pouring it into.


Understand "pour [something] in/into/on/onto
[something]" as pouring it into. Understand "empty
[something] into [something]" as pouring it into.


Pouring it into is an action applying to two things.


Check pouring it into:
    if the noun is not a fluid container, say "You can't pour
[the noun]." instead;
    if the second noun is not a fluid container, say "You
can't pour liquids into [the second noun]." instead;
    if the noun is the second noun, say "You can hardly
pour [the noun] into itself." instead;
    if the liquid of the noun is not the liquid of the second
noun:
        if the second noun is empty, now the liquid of the
second noun is the liquid of the noun;
        otherwise say "Mixing [the liquid of the noun] with
[the liquid of the second noun] would give unsavory
results." instead;
    if the noun is empty, say "No more [liquid of the noun]
remains in [the noun]." instead;
    if the second noun is full, say "[The second noun]
cannot contain any more than it already holds." instead.
```

```
Carry out pouring it into:
    let available capacity be the fluid capacity of the
second noun minus the current volume of the second
noun;
    if the available capacity is greater than the current
volume of the noun, now the available capacity is the
current volume of the noun;
    increase the current volume of the second noun by
available capacity;
    decrease the current volume of the noun by available
capacity.
```

```
Report pouring it into:
    say "[if the noun is empty][The noun] is now empty;
[otherwise][The noun] now contains [current volume of
the noun in rough terms] of [liquid of the noun]; [end if]";
    say "[the second noun] contains [current volume of
the second noun in rough terms] of [liquid of the second
noun][if the second noun is full], and is now full[end if]."
```

```
Understand the liquid property as describing a fluid
container. Understand "of" as a fluid container.
```

And now we add our liquid source kind, which will represent lakes, absinthe fountains, and any other infinite supplies of liquid we might need. Note that 3276.7 is the largest possible number of fluid ounces available to us.

```
A liquid source is a kind of fluid container. A liquid
source has a liquid. A liquid source is usually scenery.
The fluid capacity of a liquid source is usually 3276.7 fl
oz. The current volume of a liquid source is usually
3276.7 fl oz. Instead of examining a liquid source: say
"[The noun] is full of [liquid of the noun]."
```

```
Carry out pouring a liquid source into something: now
the current volume of the noun is 3276.7 fl oz.
```

We want filling things from liquid sources to work the same way as usual,

with the distinction that a) the liquid source never depletes in quantity (hence the carry-out rule resetting its fullness); and b) we should report the results a bit differently as well:

```
After pouring a liquid source into a fluid container:
    say "You fill [the second noun] up with [liquid of the
noun] from [the noun]."
```

On the other hand, pouring liquids into a liquid source needs to work completely differently from pouring liquids into anything else. Let's say we're going to allow any liquid at all to be dumped into rivers and streams (environmental protections evidently are not very well-enforced in this scenario):

```
Instead of pouring a fluid container into a liquid source:
    if the noun is empty, say "[The noun] is already
empty." instead;
    now the current volume of the noun is 0.0 fl oz;
    say "You dump out [the noun] into [the second
noun]."
```

A couple of minor refinements:

```
Swimming is an action applying to nothing. Understand
"swim" or "dive" as swimming.
```

```
Instead of swimming in the presence of a liquid source:
    say "You don't feel like a dip just now."
```

```
Before inserting something into a liquid source: say
"[The noun] would get lost and never be seen again."
instead.
```

Part 3 - Scenario

The Lakeside is a room. The Lakeside swing is an enterable supporter in the Lakeside. "Here you are by the lake, enjoying a summery view."

The glass is a fluid container carried by the player. The liquid of the glass is absinthe. The current volume of the glass is 0.8 fl oz.

The pitcher is a fluid container in the Lakeside. The fluid capacity of the pitcher is 32.0 fl oz. The current volume of the pitcher is 20.0 fl oz. The liquid of the pitcher is absinthe.

The lake is a liquid source. It is in the Lakeside.

The player wears a bathing outfit. The description of the bathing outfit is "Stylishly striped in blue and white, and daringly cut to reveal almost all of your calves, and quite a bit of upper arm, as well. You had a moral struggle, purchasing it; but mercifully the lakeshore is sufficiently secluded that no one can see you in this immodest apparel."

Instead of taking off the outfit: say "What odd ideas come into your head sometimes!"

Test me with "fill glass / empty absinthe into lake / fill glass / swim / drink lake / drink / x water / x lake".

ExamplePizza Prince

Providing a pizza buffet from which the player can take as many pieces as he wants.

Suppose we want the player to have a pizza buffet from which he can take a number of slices. But we don't want to actually put the slices there in front of him, because "you can see 17 slices of pizza here" is not the descriptive effect we want, and because we want to pretend, at least, that the pizza supply is nearly infinite. In fact, we're going to replenish the supply by allowing eaten slices to return to the buffet table (safer in IF than in real life).

To do this, we create one object to stand in for the pizza supply, but whenever the player tries to take it, we give him a different "pizza slice" object instead. Thus:

```
"Pizza Prince"

The Pizza Prince is a room.

The buffet table is a supporter in Pizza Prince.

The pizza selection is a thing on the buffet table.
Understand "slice" as the pizza selection. The
description is "They are all cheese-only, and all luke-
warm."

Rule for writing a paragraph about the buffet table:
    say "On [the buffet table] is [a pizza selection].
[description of the pizza selection][line break]".
```

Now we introduce our actual pizza slices, which are retained in a container out of play until they're needed:

```
A pizza slice is a kind of thing. 10 pizza slices are in
Pizza Limbo. A pizza slice is always edible. [After a
fashion, anyway.]
```

In this example we've set that supply to be artificially small, to make it easier to test what happens when the player reaches the limit; but we

could provide many more slices to start with in Pizza Limbo, and the aim in practice would be to pick a number high enough (such as 50 or 100) that the average player will get bored of TAKE PIZZA long before he reaches the limit.

The main thing to be aware of is that objects consume memory in the game file, so creating a large number of pizza slices might bulk the game out. This is more of a concern if we're compiling for the Z-machine than if we're compiling for Glulx.

Whenever the player tries to take the selection, we want him to wind up holding an individual slice instead; but of course we need to check and make sure that he hasn't exhausted the pizza slice supply.

```
Instead of taking the pizza selection:
    let chosen slice be a random pizza slice in Pizza
Limbo;
    if chosen slice is nothing: [That is, there were no
slices remaining]
        say "[manager refusal]";
    otherwise:
        move the chosen slice to the player;
        say "Taken (gingerly)."



To say manager refusal:
    say "[one of]'Hey!' barks a hitherto-unseen manager
from behind you. 'It's an 'all you can eat' buffet, not an
'all you can stuff down your pants' buffet.'[or]You are
conscious of a disapproving huff from the manager, so
you refrain.[stopping]"
```

That's fine for the case where the player is taking a new slice of pizza explicitly, but we need to handle it a little differently if the taking action is generated in response to EAT PIZZA. In that case, we need to take the slice and also change the identity of the noun, because after the implicit take action happens, the game will test whether the player is holding the noun before attempting to eat it. So we need to refocus its attention:

```
Rule for implicitly taking the pizza selection:
    let chosen slice be a random pizza slice in Pizza
Limbo;
    if chosen slice is nothing: [That is, there were no
slices remaining]
```

```
        say "[manager refusal]";
    otherwise:
        move the chosen slice to the player;
        say "(helping yourself from the selection)";
        now the noun is the chosen slice.
```

And finally, a bit of touch-up:

```
Rule for clarifying the parser's choice of the pizza
selection while taking:
    say "(from the magnificent selection before you)[line
break]"
```

For tidiness, we should probably also return the consumed pizza slices to Pizza Limbo so that they can be re-used later:

```
After eating a pizza slice:
    move the noun to Pizza Limbo;
    continue the action.
```

```
Test me with "i / get pizza / g / i / get pizza / drop pizza /
look / get pizza / g / look / eat pizza / g / g / g / g / get
pizza / g / g / g / g / g / g / g / g / g / g / g / g / g / i / eat pizza
/ take pizza / g".
```

ExampleExtra Supplies

A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

Suppose we have a supply closet in our game from which the player is allowed to take red pens. To keep modeling simple, we only allow him to have one in play at a time, and we test this by seeing whether the red pen is "off-stage" before moving it to his possession.

This approach might seem no different from having a single red pen sitting in the closet, but it may be preferable, for two reasons. First, it's not very plausible for a supply closet to contain nothing but a single red pen (well, assuming a well-regulated supplier, anyway); and second, it

gives the player a way to get a new red pen should the original be destroyed in a tragic handwriting accident.

"Extra Supplies"

The Supply Closet is a room. A supply of red pens is in the Supply Closet. Understand "pen" as the supply of red pens when the red pen is not visible.

There is a red pen.

Instead of taking the supply of red pens:
    if the red pen is off-stage:
        move the red pen to the player;
        say "You help yourself to a fresh red pen.";
    otherwise:
        say "You're only allowed one pen at a time. The department secretary is very strict."

South of the Supply Closet is the Furnace Room. The incinerator is a thing in the Furnace Room. It is a container. "The incinerator is here, working full blast."

After inserting something into the incinerator:
    now the noun is nowhere;
    say "A fiery blast consumes [the noun]!"

Test me with "get pen / i / get pen / get supply / s / put pen in incinerator / n / get pen".

In emulation of a certain annoying aspect of the original Adventure, in which there is a Ming vase that cannot safely be dropped:

```
"Ming Vase"

A thing can be strong or fragile. A thing is usually
strong.

Instead of attacking or dropping a fragile thing:
    now the noun is nowhere;
    say "[The noun] breaks into thousands of pieces!"

The Cave is a room. The Ming vase is carried by the
player. The vase is fragile.
```

We could also implement an additional refinement from Adventure, that a fragile thing is safe if dropped when there is a cushion nearby.

```
The pillow is a portable supporter. It is carried by the
player.

Instead of dropping a fragile thing when the pillow is in
the location: try putting the noun on the pillow instead.

After putting a fragile thing on the pillow:
    say "You set [the noun] down gently on the pillow."
```

```
Test me with "drop pillow / drop vase / get vase / get
pillow / drop vase".
```

In this rule, the pillow is the second noun; if we had a general rule about setting fragile things on soft things, we could say "You set [the noun] down gently on [the second noun]." for the same effect.

If we wanted to be more refined, we would provide extra code so that breaking a container or a supporter would leave behind their contents. We will see how to do that later.

---

123 **Example** **Spring Cleaning** *WI*

A character who sulks over objects that the player has broken (and which are now off-stage).

Here we have a destruction action that allows the player to break any fragile items. Once destroyed, these things are removed from play, but we can still refer to them: they are now off-stage. This makes it easy for our sulking character to list the ones that have been destroyed:

```
"Spring Cleaning"


A thing can be tough or fragile. A thing is usually tough.


Instead of attacking something fragile:
    say "You smash [the noun] to smithereens!";
    now the noun is nowhere.


A knick-knack is a kind of thing which is fragile.


Every turn when a knick-knack is off-stage and Granny
Blue can see the player:
    say "'Ohh,' whimpers Granny to herself softly. 'How I
will miss [the list of off-stage knick-knacks]!'"
```

```
The Parlor is a room. Granny Blue is a woman in the
Parlor. A china lamb, a porcelain milkmaid, a frolicking
Dutch cow, and a crystal unicorn are knick-knacks in the
Parlor.


Test me with "break lamb / break milkmaid / break cow /
break unicorn".
```

---

195 ExamplePaddington

WI

A CUT [something] WITH [something] command which acts differently on different types of objects.

Suppose we intend a game in which the player needs to cut things open on a regular basis. We'll want to check whether he has the proper tools handy, and deal graciously with commands such as CUT [something] when no tool is specified. So:

```
"Paddington"


A blade is a kind of thing.


Understand "cut [something] with [something]" as
cutting it with.


Instead of cutting something:
    if a blade (called the edge) is held by the player,
        try cutting the noun with the edge;
    otherwise say "Your fingernails are not sharp
enough."
```

```
Cutting it with is an action applying to two things.


Check cutting it with:
    if the noun is a person, say "That would hurt."
instead;
    if the second noun is not a blade, say "[The second
noun] has not got enough of a blade." instead.


Carry out cutting it with:
    increment the count of rips of the noun.


Report cutting it with:
    say "You slash [the noun] with [the second noun]."
```

We'll need a way to account for all these cuts and rips.

```
Definition: a thing is ripped if the count of rips of it > 0. A
thing has a number called the count of rips. After
examining something ripped, say "You see [the count of
rips of the noun in words] rip[s] in [the noun][if
something is in the noun], revealing [a list of things in
the noun][end if]."
```

Moreover, because open containers normally list their contents when examined but we'd prefer Paddington's to be mentioned in the ripping paragraph:

```
The examine containers rule does nothing when
examining the teddy bear.
```

So far, so good. But suppose that we'd like cutting also to make containers be permanently open and impossible to close again. We could write an "instead" rule, but that would mean that only our instead instructions would take effect, overriding the normal cutting it with rules entirely. Better would be to add a second carry out rule:

```
Carry out cutting a container with something:
    now the noun is open;
    now the noun is unopenable.
```

Now our rule will occur whenever a container is cut, but play will still go on to the reporting stage. And indeed we can add more of these, of varying degrees of specificity:

```
Carry out cutting something which is part of something
with something:
    move the noun to the player.
```

```
Carry out cutting the quilt with something:
    now the description of the quilt is "Horribly tattered."
```

For that matter, we might want to add a report rule as well, to occur after the "You slash..." rule, so that every time the player cuts something open which has contents, the contents will be listed.

```
Report cutting it with:
    if the noun is open and the noun contains something,
        say "Visible within [is-are a list of things in the
noun]."
```

This time we do not add the condition to the rule (i.e., Report cutting an open noun...) If we did, this report rule would be more specific than the general report rule, and would occur first.

```
The Safehouse is a room.
```

```
The teddy bear is a closed thing in the Safehouse. The
description is "Fluffy[if the head is part of the bear], with
an outsized head[otherwise], but headless[end if]." The
head is a closed part of the teddy bear. In the bear is a
large wad of stuffing. In the head are a small wad of
stuffing and a packet of smuggled diamonds.
```

```
The quilt is in the Safehouse. The description is "An old
but comforting quilt."


The player carries a blade called a switchblade.
```

Here is a final nicety to get rid of the "which is closed" statement on our closed unopenable teddy bear, using an "activity" rule:

```
After printing the name of a closed unopenable
container:
    omit contents in listing.


Test me with "cut quilt with bear / cut quilt with
switchblade / examine quilt / cut bear with switchblade /
again / examine bear / cut head with switchblade / get
diamonds / mourn loss of innocence".
```

---

314 **ExampleTerracottissima** *WI*

The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

This easiest way to do this uses the "printing the name of" activity, which will come up in the following chapter:

```
"Terracottissima"


A flowerpot is a kind of thing. Understand "pot" as a
flowerpot.


A flowerpot can be unbroken or broken. After dropping
an unbroken flowerpot: say "Crack!"; now the noun is
```

broken. Understand the broken property as describing a flowerpot.

Before printing the name of a broken flowerpot, say "broken ". Before printing the name of an unbroken flowerpot: if a flowerpot is broken, say "unbroken ".

Before printing the plural name of a broken flowerpot, say "broken ". Before printing the plural name of an unbroken flowerpot: if a flowerpot is broken, say "unbroken ".

The Herb Garden is a room. In the Herb Garden are ten unbroken flowerpots.

Test me with "get three flowerpots / drop all / look".

ExampleKyoto

Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

Suppose we want to expand the function of the existing THROW SOMETHING AT command so that a thrown object actually does make contact most of the time. A glance at the Actions index tells us that the Throwing it at rulebook currently looks like this:

Throwing something at something (past tense thrown it at)
"drop [something held] at/against/on/onto [something]"

[illegible] throwing something at rule
[illegible] throwing something at rule
[illegible] throwing something at

Some of those still look useful. We want to leave the "implicitly remove thrown clothing" rule, for instance -- no fair having the player throw a hat that's on his head. On the other hand, the "futile to throw things at inanimate objects rule" is going to have to go, because that would prevent us from ever being able to complete the throwing command. So let's get rid of that:

```
"Kyoto"

Part 1 - Throwing Rules

The futile to throw things at inanimate objects rule is not
listed in the check throwing it at rules.
```

That "block throwing at" rule also looks sinister: any "block..." rule in the standard actions library is there to print a message telling the player he can't do what he's asked to do.

But it's not enough to ignore it, the way we did the "futile" rule. Since we are only expanding the command to affect inanimate objects, let's replace the "block throwing at" rule with a different one which will only prevent the player throwing things at people:

```
The block throwing at people rule is listed instead of the
block throwing at rule in the check throwing it at rules.

This is the block throwing at people rule:
    if the second noun is a person, say "That might be
construed as an attack." instead.
```

Now we've changed the command so that some action can sometimes be carried out here -- but we don't have any rules for what happens. It's time to create some rules for our model world.

```
A thing can be hard or soft. A thing can be fragile or
```

```
strong. Shape is a kind of value. The shapes are round,
flat, and linear. A thing has shape.
```

If we're actually going to allow throwing, we might want to add a couple of extra checks to the rulebook to make sure that this happens when it ought:

```
Check throwing it at (this is the block juggling rule):
    if the player is carrying the second noun, say "It
would be difficult to throw at something you are yourself
holding." instead.
```

```
Check throwing it at (this is the avoid throwing things
into themselves rule):
    if the second noun is within the noun, say "That would
be a nice magic trick." instead.
```

And then the rules for the action itself:

```
Carry out throwing it at (this is the check aerodynamics
rule):
    if the noun is flat:
        move noun to location;
        say "[The noun], unwieldy, flutters to the ground.";
        rule succeeds.
```

That "rule succeeds" ends the action here, if the noun is flat. If not, Inform goes on to the next rule in the carry out throwing it at rulebook:

```
Carry out throwing it at (this is the contact rule):
    say "[The noun] hits [the second noun].[paragraph
break]";
    if the second noun is fragile and the noun is hard:
        destroy the second noun.
```

```
Carry out throwing it at (this is the landing rule):
    let destination be the location;
    if the second noun is on a supporter (called endtable),
let destination be the endtable;
```

```
    if the second noun is a supporter, let destination be
the second noun;
    move the noun to the destination;
    if the noun is fragile and the second noun is hard:
        destroy the noun;
        rule succeeds;
    say "[The noun] lands [if the destination is the
location]nearby[otherwise]on [the destination][end if]."
```

These rules are assuming some backup information, so let's provide that as well:

```
Reliance relates a thing (called X) to a thing (called Y)
when X is part of Y or X is in Y or X is on Y. The verb to
be relying on means the reliance relation.
```

```
To destroy (item - a thing):
    let home be the holder of the item;
    if the item is part of something (called the
superstructure), let home be the holder of the
superstructure;
    if the item is visible:
        say "[The item] breaks[if something is relying on
the item], leaving [a list of things which are relying on
the item] behind[end if].";
    if something is relying on the item,
        now all the things which are relying on the item are
in the home;
    now the item is nowhere.
```

Now suppose we'd like to add some further cases for what happens if the player breaks a fragile door this way:

```
To destroy (item - a door):
    now the item is open;
    now the item is unopenable;
    say "[The item] smashes."
```

```
Rule for printing the name of an unopenable open door
while not throwing something at something:
    say "open doorway".
```

```
Understand "door" or "doorway" as a door.
```

This works, except that objects will continue to "strike" open, unopenable doors, with the result that the player can smash the same door over and over. What we need is another rule, after the aerodynamics rule and before the contact rule, that tells Inform how to handle throwing things at open doors.

```
This is the flying through doorways rule:
    if the second noun is an open door:
        let the distant room be the other side of the second
noun;
        move the noun to the distant room;
        say "[The noun] flies out of sight into [the distant
room].";
        rule succeeds.
```

If the original rulebook is one we wrote ourselves, we could just add that rule in the proper spot in order. If we got it from an extension, though, we might need to put it in the right place explicitly:

```
The flying through doorways rule is listed before the
contact rule in the carry out throwing it at rules.
```

The magic of rulebooks is that they allow authors to amend each other's work (or the Standard Rules) with a fair amount of freedom. A well-written extension will give individual names to its rules, to allow subsequent authors to modify the function of the extension without too much trouble.

Now for an actual scenario with which to test this:

```
Part 2 - The Study
```

```
The sliding paper screen is a door. It is north of the
Moss Garden and south of the Study. The paper screen
is fragile.
```

The player carries a netsuke and a shamisen. The description of the netsuke is "A weight for the cord on which you wear your purse or your medicine box. This particular one has the shape of a bullfrog, carved from green stone." The netsuke is round, hard, and strong. Understand "green" or "stone" or "bullfrog" as the netsuke.

The description of the shamisen is "An instrument you have only begun to learn to play." The shamisen is linear, soft, and fragile. A neck is part of the shamisen. The neck is linear, strong, and hard. A body is part of the shamisen. The body is round, fragile, and soft. A string is part of the shamisen. The string is linear, soft, and strong. The printed name of the body is "[if the body is not part of the shamisen]shamisen [end if]body". The printed name of the neck is "[if the neck is not part of the shamisen]shamisen [end if]neck". Understand "shamisen" as the body when the body is not part of the shamisen. Understand "shamisen" as the neck when the neck is not part of the shamisen.

The description of the Study is "A restful three-tatami room." The Study contains a calligraphy box and a hanging scroll. The initial appearance of the hanging scroll is "A handsome scroll depicts two women in kimonos crossing a bridge; Mount Fuji is in the background." The calligraphy box contains a brush. The box is openable and closed. The brush is hard, linear, and strong. The calligraphy box is round, soft, and strong. The hanging scroll is flat, soft, and strong.

The description of the Moss Garden is "Earlier today, you arranged three leaves on the moss in imitation of autumn. They must not be disturbed." The leaves are scenery in the Moss Garden. Instead of throwing something at the leaves: say "You spent too long over their placement."

Test me with "test one / test two".

Test one with "open screen / throw netsuke at screen / n / get netsuke / close screen / get scroll / throw scroll at screen / throw netsuke at scroll / get netsuke / throw netsuke at shamisen / drop netsuke".

Test two with "throw shamisen at netsuke / get all / throw netsuke at screen / get netsuke / throw netsuke at door / s / get netsuke".

Example **Swerve left? Swerve right? Or think about it and die?**

Building a marble chute track in which a dropped marble will automatically roll downhill.

WI

Suppose we have marbles that roll downhill across our map, in a life-size version of one of those marble-chute toys. We might now want to keep track of both compass relationships and which-room-slopes-into-which, so we make a new relation:

"Swerve left? Swerve right? Or think about it and die?"

Overlooking relates various rooms to various rooms.

The verb to overlook means the overlooking relation.

A thing can be spherical or lumpy. A marble is a kind of thing. A marble is always spherical. The player carries a marble called a red marble. The player carries a marble called an agate marble. The player carries a marble called a blue cloudy marble.

```
The Long Yellow Slide is north of the Funnel. The Long
Yellow Slide overlooks the Blue Funnel. The Ski-jump is
below the Blue Funnel. The Blue Funnel overlooks the
Ski-jump. The Ski-jump overlooks the Landing Bowl.
The Landing Bowl overlooks the Snake Run. The
Landing Bowl is north of the Snake Run. The Snake
Run overlooks the Goal. The Snake Run is north of the
Goal.


Definition: a room is sloping if it overlooks a room.
```

And let's say we want the player to be allowed to slide, too, since that would be much more fun than just watching the marbles go:

```
Understand "sit" as sitting down. Sitting down is an
action applying to nothing. Check sitting down: if the
player is spherical, say "You are already seated." Carry
out sitting down: now the player is spherical. Report
sitting down: say "You sit, ready to slide wherever fate
takes you."


Understand the command "stand" as something new.


Understand "stand" or "stand up" as standing up.
Standing up is an action applying to nothing. Check
standing up: if the player is lumpy, say "You are already
standing." Carry out standing up: now the player is
lumpy. Report standing up: say "You get to your feet."
```

Now a rule to control what happens to all our sliding and rolling objects:

```
Every turn:
   repeat with item running through spherical things
which are in sloping rooms:
      let the current space be the holder of the item;
      let the final space be a random room which is
overlooked by the current space;
```

```
        if the player can see the item and the item is a
    marble, say "[The item] rolls out of the room toward [the
    final space].[line break]";
        if the player is the item, say "You keep sliding...";
        move the item to the final space;
        if the player can see the item and the item is a
    marble, say "[The item] rolls into the room from [the
    current space].[line break]".
```

Since the Ski-jump overlooks the Landing Bowl, the marble will be able to fly through the air to its destination, even though there is no map connection to allow the player to cross. We might want to let the player make it across this barrier also, so:

```
Instead of jumping in a sloping room:
    say "You leap...";
    move the player to a random room overlooked by the
location.
```

Because overlooking is various-to-various, we could include that element popular in marble chute toys, the splitter:

```
The Downhill Splitter is north of the Long Yellow Slide.
"The green plastic chute runs downhill towards a Y-
junction, forcing incoming marbles right or left."
```

```
The Downhill Splitter overlooks the Long Yellow Slide
and the Purple Snaking Passage. The Purple Snaking
Passage is southeast of the Downhill Splitter. The
Purple Snaking Passage overlooks the Landing Bowl.
The Purple Snaking Passage is above the Landing
Bowl.
```

```
The player is in the Downhill Splitter.
```

```
Test me with "drop red / drop blue / sit / z / stand up /
drop agate / sit / z / z / z / z / z".
```

ExampleDepth

Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

In the following, we pretend that every item has a cuboidal shape. Every thing has a length, width and depth, while a "measured container" also has interior dimensions. (Thus a 10x10x10 container with 1cm-thick sides might have interior dimensions 9x9x9.)

"Depth"

A length is a kind of value. 10 cm specifies a length. An area is a kind of value. 10 sq cm specifies an area. A length times a length specifies an area. A volume is a kind of value. 10 cu cm specifies a volume. A length times an area specifies a volume.

A thing has a length called height. A thing has a length called width. A thing has a length called depth. The height of a thing is usually 10 cm. The width of a thing is usually 10 cm. The depth of a thing is usually 10 cm.

```
To decide what volume is the exterior volume of (item -
a thing):
    let base area be the height of the item multiplied by
the width of the item;
    let base volume be the base area multiplied by the
depth of the item;
    decide on the base volume.
```

In order to see how these shapes might fit together spatially, we need to work out the three dimensions in order of size. (If we were only dealing with portable objects, we could simply insist that the length always be greater than the width which in turn must be greater than the depth, because we could always turn them over in our hands until this was so: but some of the things we deal with may be fixed in place.) A clever way to do this might be to put them in a table of three rows and sort it, but we

will write the calculation out longhand:

```
To decide what length is the largest dimension of (item -
a thing):
    let long side be the height of item;
    if the width of the item is greater than the long side,
now the long side is the width of the item;
    if the depth of the item is greater than the long side,
now the long side is the depth of the item;
    decide on the long side.


To decide what length is the middling dimension of (item
- a thing):
    let longer side be the height of item;
    let shorter side be the width of item;
    if the width of the item is greater than the height of the
item:
        let shorter side be the height of item;
        let longer side be the width of item;
    if the depth of the item is greater than the longer side,
decide on the longer side;
    if the depth of the item is less than the shorter side,
decide on the shorter side;
    decide on the depth of the item.


To decide what length is the shortest dimension of (item
- a thing):
    let short side be the height of item;
    if the width of the item is less than the short side, now
the short side is the width of the item;
    if the depth of the item is less than the short side, now
the short side is the depth of the item;
    decide on the short side.
```

When testing this example, the author made use of the following: it's no longer needed, but may be useful to anyone else planning elaborations.

```
To test the dimensions of (item - a thing):
    say "[the item] - height [height of the item], width
[width of the item], depth [depth of the item].";
    say "largest side [largest dimension of the item],
middling [middling dimension of the item], smallest
```

```
        [shortest dimension of the item]."
```

We now introduce a new kind: a measured container, which not only has exterior dimensions - the height, width and depth which every thing now has - but also interior measurements. A convenient way to do calculations with the hollow interior is to regard it as if it were a solid shape in its own right, and we do this with the aid of something out of world, which the player never sees: the "imaginary cuboid", which is made into the shape of whatever measured container's interior is being thought about.

```
        A measured container is a kind of container. A
        measured container has a length called interior height.
        A measured container has a length called interior width.
        A measured container has a length called interior depth.


        There is an imaginary cuboid.


        To imagine the interior of (receptacle - a measured
        container) as a cuboid:
            now the height of the imaginary cuboid is the interior
        height of the receptacle;
            now the width of the imaginary cuboid is the interior
        width of the receptacle;
            now the depth of the imaginary cuboid is the interior
        depth of the receptacle.


        To decide what volume is the interior volume of
        (receptacle - a measured container):
            imagine the interior of the receptacle as a cuboid;
            decide on the exterior volume of the imaginary
        cuboid.
```

If we assume that we could always pack items into a measured container with perfect ease, never wasting any space, then the only volume constraint will be that the total volume of the contents must not exceed the volume of the inside of the container. So we need to calculate the available volume.

```
To decide what volume is the available volume of
(receptacle - a measured container):
    let the remaining space be the interior volume of the
receptacle;
    repeat with item running through things in the
receptacle:
        decrease the remaining space by the exterior
volume of the item;
    if the remaining space is less than 0 cu cm, decide on
0 cu cm;
    decide on the remaining space.
```

If we only constrained volume, a 140 cm-long fishing rod could fit into a 12 cm by 12 cm compact disc box. So we also insist the basic shape must fit, in some orientation perpendicular to one of the sides (i.e.: we can turn the item over in any of its three sides, but not turn it diagonally or wedge it in at a tilt). This requires the longest side of the item to be less than the longest side of the receptacle, and the middle-length side, and also the shortest side. The number of these conditions to fail gives us a clue as to how we can best describe the reason why the shape won't squeeze in.

```
Check inserting something (called the item) into a
measured container (called the receptacle):
    if the exterior volume of the item is greater than the
interior volume of the receptacle, say "[The item] will
never fit inside [the receptacle]." instead;
    if the exterior volume of the item is greater than the
available volume of the receptacle, say "[The item] will
not fit into [the receptacle] with [the list of things in the
receptacle]." instead;
    imagine the interior of the receptacle as a cuboid;
    if the largest dimension of the item is greater than the
largest dimension of the imaginary cuboid, say "[The
item] is too long to fit into [the receptacle]." instead;
    if the middling dimension of the item is greater than
the middling dimension of the imaginary cuboid, say
"[The item] is too wide to fit into [the receptacle]."
instead;
    if the shortest dimension of the item is greater than
the shortest dimension of the imaginary cuboid, say
"[The item] is too bulky to fit into [the receptacle]."
instead.
```

And finally a situation to try out these rules.

```
The Cubist Lab is a room. "A laboratory which, as the
art critic Louis Vauxcelles said about Braque's paintings
in 1908, is full of little cubes: everyday objects rendered
as if cuboidal."


The box is a measured container. The interior height is
10 cm. The interior depth is 5 cm. The interior width is 6
cm. The player carries the box.


A pebble is a kind of thing. The height is usually 2 cm.
The depth is usually 2 cm. The width is usually 2 cm.
The player carries 25 pebbles.


A red rubber ball is carried by the player. The depth is 5
cm. The width is 5 cm. The height is 5 cm.
An arrow is carried by the player. The height is 40 cm.
The width is 1 cm. The depth is 1 cm.
A crusty baguette is carried by the player. The height is
80 cm. The width is 4 cm. The depth is 5 cm.
A child's book is carried by the player. The height is 1
cm. The width is 9 cm. The depth is 9 cm.
A featureless white cube is carried by the player. The
height is 6 cm. The width is 6 cm. The depth is 6 cm.


Test me with "put arrow in box / put book in box / put
cube in box / put ball in box / put baguette in box / put
pebbles in box".
```

Several warnings about this. First, the numbers can't go very high (if the Settings for the project set the story file format to the Z-machine): while the volume can in theory go to 32,767, in practice this equates to an object 32 cm on a side, which is not very large. One way to avoid this is to use the Glulx format, allowing for sizes in excess of 10 m on a side: or we could simply scale the dimensions to suit our purposes, using a decimeter (10 cm) as the basic unit of measurement, for instance.

Second, the system will require a height, width, and depth for every portable object in the game, which is a large commitment to data entry; it may become tiresome. So it is probably not worth bothering with this kind of simulation unless it is going to be genuinely significant.

This example draws together the previous snippets into a working implementation of the weighbridge.

The following is not a very sophisticated approach, because it does not allow for weight to accumulate: if we put a gold ingot into a paper bag, then put the bag on the balance platform, only the bag's weight will register. But it will do for a first try.

"Dimensions"

A length is a kind of value. 10m specifies a length. An area is a kind of value. 10 sq m specifies an area. A length times a length specifies an area.

A weight is a kind of value. 10kg specifies a weight. Everything has a weight.

The verb to weigh means the weight property. A thing usually weighs 1kg.

Definition: A thing is light if its weight is 3kg or less.

Definition: A thing is heavy if its weight is 10kg or more.

The Weighbridge is a room.

A blackboard is in the Weighbridge. "A blackboard propped against one wall reads: '122/10 is [122 divided by 10] remainder [remainder after dividing 122 by 10]; 122kg/10kg is [122kg divided by 10kg] remainder [remainder after dividing 122kg by 10kg]; 122kg/10 is [122kg divided by 10] remainder [remainder after dividing 122kg by 10].'" The blackboard weighs 10kg.

A feather and a lead pig are in the Weighbridge. The lead pig weighs 45kg.

The balance platform is a supporter in the Weighbridge. "The balance platform is 10m by 8m, giving it an area of [10m multiplied by 8m], currently weighing [the list of things on the platform]. The scale alongside reads: [total weight of things on the platform]. [if two things are on the platform]Average weight is: [the total weight of things on the platform divided by the number of things on the platform]. Heaviest item is [the heaviest thing on the platform], at [weight of the heaviest thing on the platform]. Lightest item is [the lightest thing on the platform], at [weight of the lightest thing on the platform]. [otherwise]It seems to be able to weigh several things at once."

Test me with "get feather / put it on platform / look / get pig / put it on platform / look".

---

270 **Example** **The Speed of Thought**

Describing scientifically-measured objects in units more familiar to the casual audience. *WI*

Suppose that we have a number of objects in the game that are sized in some conventional unit (such as meters), but which we would like to describe in slightly less formal terms. To do this, we will start with measurements as defined in the built-in extension Metric Units, so we don't have to recreate all these.

We'll add our own set of "conceptual units" -- things we're familiar with in

real life. As we'll see below, Inform will automatically choose a unit of the right order to express a given distance if we tell it to print a length "in conceptual units".

Note: the following will compile only if you have settings set for Glulx. (To change this, go to the Settings panel and click on the Glulx option.) The Glulx virtual machine is capable of handling larger numbers than the Z-machine.

"The Speed of Thought"

Section 1 - Procedure

Include Metric Units by Graham Nelson.

1 quarter (in conceptual units, in quarters, singular) or 2 quarters (in conceptual units, in quarters, plural) specifies a length equivalent to 24mm.
1 pencil (in conceptual units, in pencils, singular) or 2 pencils (in conceptual units, in pencils, plural) specifies a length equivalent to 18cm.
1 bathtub (in conceptual units, in bathtubs, singular) or 2 bathtubs (in conceptual units, in bathtubs, plural) specifies a length equivalent to 152cm.
1 Olympic swimming pool (in conceptual units, in Olympic swimming pools, singular) or 2 Olympic swimming pools (in conceptual units, in Olympic swimming pools, plural) specifies a length equivalent to 50 meters.
1 Empire state building (in conceptual units, in Empire State buildings, singular) or 2 Empire State buildings (in conceptual units, in Empire State buildings, plural) specifies a length equivalent to 443m.

1 credit card (in conceptual units, in credit cards, singular) or 2 credit cards (in conceptual units, in credit cards, plural) specifies an area equivalent to 46 sq cm.
1 letter sheet (in conceptual units, in letter sheets, singular) or 2 letter sheets (in conceptual units, in letter

sheets, plural) specifies an area equivalent to 603 sq cm.
1 queen-sized mattress (in conceptual units, in queen-sized mattresses, singular) or 2 queen-sized mattresses (in conceptual units, in queen-sized mattresses, plural) specifies an area equivalent to 3 square meters.
1 football field (in conceptual units, in football fields, singular) or 2 football fields (in conceptual units, in football fields, plural) specifies an area equivalent to 5351 square meters.

Understand "report [something]" as reporting. Reporting is an action applying to one thing.

Check reporting:
    if the noun is not a fact:
        say "The public doesn't want to hear about [the noun]." instead.

Carry out reporting:
    now the noun is nowhere.

Report reporting:
    if the extent of the noun is greater than 0mm and the surface of the noun is greater than 0 sq cm:
        contextualize "'[The noun] has a length of [about] [extent of the noun in conceptual units] and an area of [about] [surface of the noun in conceptual units].'";
    otherwise if the extent of the noun is greater than 0mm:
        contextualize "'[The noun] has a length of [about] [extent of the noun in conceptual units].'";
    otherwise if the surface of the noun is greater than 0 sq cm:
        contextualize "'[The noun] has an area of [about] [surface of the noun in conceptual units].'";
    otherwise:
        say "'[The noun] is... pretty hard to imagine,' you say weakly. That's not going to go over well."

To say about:
    say "[one of]roughly[or]about[or]around[or]approximately[at random]";

To contextualize (chosen information - text):
    say "[one of]You turn to the camera and speak:[or][or]Turning to another camera angle, you add:[or][stopping] ";
    say "[chosen information] ";
    say "[one of][line break][or]Right now the station will be cutting over to a visual of that.[or][line break][or]Pity the kids in audiovisual who have to scare that image together in a hurry.[or]You smile brightly.[stopping]";

Section 2 - Scenario

The Science Journalism Desk is a room. "From here you, the Science Anchor, have the privilege of reporting the latest and most fascinating stories to an eager public."

After looking:
    try thinking.

Instead of thinking:
    say "Currently you have to report on the International Space Station. Your story could include [the list of facts carried by the player]."

Instead of taking inventory:
    say "It looks foolish to be fiddling with your possessions on camera."

```
Instead of dropping a fact:
    say "You decide to omit [the noun] from your lineup.";
    now the noun is nowhere.



A fact is a kind of thing. Every fact is carried by the
player. A fact has a length called the extent. A fact has
an area called the surface.



The experiment module is a fact. The extent is 1116cm.
The logistics module is a fact. The extent is 421cm.
The solar array is a fact. The surface is 375 sq m. The
extent is 58m.
An individual solar cell is a fact. The surface is 8 sq cm.
The orbit height is a fact.



Report reporting the orbit height:
    contextualize "'The station orbits at heights between
[about] [278km in conceptual units] and [460km in
conceptual units] above the earth.'" instead.



Every turn:
    if the player carries no facts:
        say "And that's all! The channel cuts to weather.";
        end the story saying "Time for lunch".



Test me with "report experiment module / report
logistics / report height / report array / report solar cell".
```

ExampleLead Cuts Paper

To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster.

The following shows the kind of "realism" rules which could be introduced using weights. Not entirely realistic: we do not bother to rupture

containers out of the player's sight.

"Lead Cuts Paper"

A weight is a kind of value. 10kg specifies a weight. Everything has a weight. A thing usually has weight 1kg.

A container has a weight called breaking strain. The breaking strain of a container is usually 50kg. Definition: A container is bursting if the total weight of things in it is greater than its breaking strain.

A lead pig, a feather, a silver coin and a paper bag are in a room called the Metallurgy Workshop. The paper bag is a container with breaking strain 2kg. The lead pig has weight 50kg.

Every turn when a container (called the sack) held by someone visible (called the chump) is bursting:
    say "[The sack] splits and breaks under the weight! [if the player is the chump]You discard[otherwise][The chump] discards[end if] its ruined remains, looking miserably down at [the list of things in the sack] on the floor.";
    now all of the things in the sack are in the location;
    now the sack is nowhere.

Test me with "get bag / get feather / put feather in bag / get pig / put pig in bag / look".

A kind of rope which can be tied to objects and used to anchor the player or drag items

from room to room.

The range of things one might want to do with a rope in a work of interactive fiction is fairly overwhelming. One might, in theory, swing from ropes; use them to tie containers shut; cut them up into smaller ropes; tie them together into longer ropes; employ them as fuses; bind other characters with them, or the player character.

Our rope implementation is, by these lights, reasonably simple, but it does account for the possibility of tying and untying both ends; using ropes to descend into lower rooms; pulling objects tied to the far end of the rope; and dragging objects from place to place.

```
"Otranto"
```

We start by coming up with a rope.

```
A rope is a kind of thing.

Definition: a thing is nonrope if it is not a rope. [The
perfect idiocy of this statement notwithstanding, having
a shortcut will come in very handy later]

Attachment relates things to each other in groups. The
verb to be stuck to means the attachment relation.

Definition: a thing is tied if the number of things stuck to
it is greater than 1.

Definition: a thing is free if it is not tied.

Definition: a rope is free if the number of nonrope things
stuck to it is less than 2.
```

```
Definition: a thing is hindering if it is stuck to the noun
and it is not within the location.
```

```
A thing can be round or unevenly shaped. A thing is
usually round.
```

```
Definition: something is anchored if it is fixed in place or
it is scenery or it is part of an anchored thing.
```

```
Definition: something is draggable if it is not had by the
player and it is not the player and it is not anchored.
```

Now, we want a rope to be described in terms of the way it is tied, when it's described in a room description.

```
Rule for writing a paragraph about a rope (called the
coil):
    if the coil is stuck to something which is in a room
(called the next room) which is not the location:
        let the way be the best route from the location to
the next room;
        if the way is up or the way is down:
            say "[The coil] runs [way] into [the next room].";
        otherwise:
            say "[The coil] snakes across the floor [way]
towards [the next room].";
    otherwise:
        say "There is [a coil] here[if the coil is stuck to a
visible nonrope thing], tied to [the list of nonrope visible
things which are stuck to the coil][end if]."
```

```
To decide what room is the home of (item - a thing):
    if item is a door:
        let front cut be the number of moves from the
location to the front side of the item;
        let back cut be the number of moves from the
location to the back side of the item;
```

```
        if front cut is -1, let front cut be 999;
        if back cut is -1, let back cut be 999;
        if the location encloses the item, decide on the
    location;
        if front cut is greater than back cut, decide on the
    back side of the item;
        decide on the front side of the item;
    decide on the location of the item.
```

```
Rule for writing a paragraph about a nonrope thing
(called the anchor) which is stuck to a rope (called the
coil):
    if the coil is in an adjacent room:
        let the next room be the home of the coil;
        let the way be the best route from the location to
    the next room;
        if the way is up or the way is down:
            say "[The coil] runs [way] from [the anchor] into
    [the next room].";
        otherwise:
            say "From [the anchor] runs [a coil], heading off
    toward [the way].";
    otherwise:
        if the coil is stuck to something which is not visible,
            say "[The coil] is tied to [the anchor][if the coil is
    stuck to something in an adjacent room (called the next
    room)], and from there runs off towards [the next room]
    [end if]."
```

We need a way to account for it when it's being carried, as well.

```
After printing the name of a rope (called the tied object)
while taking inventory:
    if something nonrope is stuck to the tied object:
        say " (attached to [the list of nonrope things which
    are stuck to the tied object])";
    otherwise:
        say " (with both ends free)".
```

And, indeed, whenever the player examines a rope, we should see what's connected.

```
Instead of examining a rope (called the cord) when
something is stuck to the cord:
```

```
    say "[The noun] is tied to [the list of secondary things
which are stuck to the noun]."
```

Similarly, any time the player looks at something tied to a rope.

```
After examining the player when the player is stuck to
something which is not the player:
    say "You're currently lashed to [the list of secondary
things stuck to the noun]."
```

```
After examining something which is stuck to something
secondary:
    say "[The noun] is currently attached to [the list of
secondary things stuck to the noun]."
```

We also need to make sure that the rope can be interacted with properly even when it's partly in the next room.

```
After deciding the scope of the player:
    if something stuck to a rope (called the coil) is in the
location, place the coil in scope.
```

```
A reaching inside rule:
    if the noun is a rope:
        let the anchor be a random visible thing stuck to
the noun;
        if the anchor is touchable, allow access.
```

Now tying:

```
Before tying something to a rope:
    if the noun is stuck to the second noun, say "[The
noun] and [the second noun] are already tied together."
instead;
    if the second noun is not free, say "[The second
noun] has no ends free." instead;
    if the noun is round, say "You can't realistically tie
anything to [the noun]." instead.
```

```
Instead of tying a rope to something:
    try tying the second noun to the noun.


Instead of tying something to a rope:
    now the noun is stuck to the second noun;
    say "You loop [the second noun] around [the noun]
and knot firmly."


Instead of tying something to a nonrope tied thing:
    let the coil be a random rope stuck to the second
noun;
    try tying the noun to the coil.


Instead of tying a nonrope tied thing to something:
    let the coil be a random rope stuck to the noun;
    try tying the second noun to the coil.


Instead of tying a free nonrope thing to a free nonrope
thing:
    if the player carries a free rope (called the coil):
        try tying the noun to the coil;
        if the noun is stuck to the coil and the coil is free:
            try tying the second noun to the coil;
    otherwise:
        say "You lack the requisite spare rope."


Understand "untie [something] from [something]" as
untying it from. Understand "untie [something]" as
untying it from.


Rule for supplying a missing second noun while untying
something from:
    if the number of secondary things stuck to the noun is
0, say "[The noun] is already entirely free." instead;
```

```
    if the noun is a rope:
        if the number of touchable nonrope things which
are stuck to the noun > 1:
            say "You'll have to say which thing you want to
untie [the noun] from.";
            rule fails;
        otherwise:
            if the number of touchable nonrope things stuck
to the noun is 0, say "You can't reach [the random
nonrope thing stuck to the noun]." instead;
            let the tied object be a random touchable
nonrope thing which is stuck to the noun;
            say "(from [the tied object])[line break]";
            now the second noun is the tied object;
    otherwise:
        if the noun is stuck to a rope (called the tied
object):
            say "(from [the tied object])[line break]";
            now the second noun is the tied object.



Untying it from is an action applying to two things.



Before untying a rope from something: try untying the
second noun from the noun instead.



Before untying something from a rope:
    if the second noun is not held:
        say "(first picking up [the second noun])[line
break]";
        try taking the second noun.



Check untying it from:
    unless the noun is stuck to the second noun or the
second noun is stuck to the noun,
        say "[The noun] and [the second noun] are already
not tied together." instead.



Carry out untying it from:
    now the noun is not stuck to the second noun.
```

```
Report untying it from:
    say "Untied."
```

Another part of the fun of a rope is that you can drag things from another room:

```
After reading a command: now every thing is
unmentioned.


Before pulling something anchored: say "[The noun] is
firmly anchored." instead.


Instead of pulling something tied:
    if the noun is unmentioned:
        say "The impulse is transmitted to [the list of
pullable things stuck to the noun].";
        repeat with item running through pullable things
stuck to the noun:
            say "[item]: [run paragraph on]";
            try pulling the item;
        if the noun is a rope and the noun is not within the
location:
            if the number of nonrope hindering things is 0,
move the noun to the location;
    otherwise:
        continue the action.


Before pulling something which is not visible:
    if the noun is anchored:
        say "[The noun] resists, for whatever reason."
instead;
    otherwise:
        let space be the holder of the noun;
        let way be the best route from the space to the
location;
        if the way is a direction:
            move the noun to the location;
            say "[The noun] [if the way is
up]rises[otherwise]slides[end if] into view." instead;
```

```
        otherwise:
            move the noun to the location;
            say "[The noun] slides into view." instead.


Definition: a thing is secondary if it is not the noun.
Definition: a thing is pullable if it is not the noun and it is
not the player.
```

A player who is tied to things should also have some restrictions on his ability to move.

```
Before going a direction (called the way) when the
player has something (called the link) which is stuck to
something anchored (called the anchor):
    let the next room be the home of the anchor;
    if the next room is not a room, continue the action;
    if the next room is the location:
        if the link is stuck to at least two anchored things,
            say "You can't go far while you're carrying [the
link] tied to [the list of anchored things stuck to the link]."
instead;
    otherwise:
        let the safe way be the best route from the location
to the next room;
        if the safe way is the way:
            if the player is not stuck to the anchor, say
"(coiling up your rope again as you go...)";
        otherwise:
            if the safe way is a direction,
                say "While you have [the link] you can't really
head any direction but [best route from the location to
the next room]." instead;
            otherwise say "You're tied up here." instead.


Before going a direction (called the way) when the
player is stuck to something anchored (called the
anchor):
    let the next room be the home of the anchor;
    if the next room is not a room, continue the action;
    if the next room is the location:
        if the player is stuck to at least two anchored
things,
            say "You can't go far while you're tied to [the list
of anchored things stuck to the player]." instead;
```

```
    otherwise:
        if the best route from the location to the next room
    is the way:
            say "(coiling up your rope again as you go...)";
        otherwise:
            say "Your attachments prevent you going any
    way but [best route from the location to the next room]."
    instead.
```

Sometimes, if the player is tied to a movable object, the moved object will move with him.

```
After going somewhere when the player has something
(called the link) which is stuck to something draggable:
    if the player is not stuck to the link:
        say "You drag along behind you [the list of
    draggable things which are stuck to the link].";
        now every draggable thing which is stuck to the link
    is in the location;
    continue the action.
```

```
Report going somewhere when the player is stuck to
something draggable:
    say "You drag along behind you [the list of draggable
    things which are stuck to the player].";
    now every draggable thing which is stuck to the
    player is in the location.
```

And now the actual game and puzzles.

```
Use full-length room descriptions.
```

```
The Fallow Field is a room. "The very land is gloomy,
the earth plowed into untended rows that yield no fruit,
shadowed by the castle to the north. A chasm, no doubt
the product of some upheaval of the earth, opens before
your feet.". An oak stump is fixed in place in the Field.
"From an oak stump, a few hopeful shoots grow." A
hempen rope is a rope in the field. It is stuck to the oak
stump and the wooden chest. The stump is unevenly
shaped.
```

The Chasm is below the Field. "Your person is most uncomfortably pressed on every side by the closeness of the walls; to which you may add as a further inconvenience, that the irregularity of the floor making it difficult to walk upright." An iron key is in the Chasm. "An iron key nestles in the cleft of earth, its age indicated by its implausibly great size."

The wooden chest is a unevenly shaped closed openable container in the Chasm. The description of the wooden chest is "A handsome, solid case not long committed to its dank enclosure, or it would long since have rotted." Rule for printing the name of the wooden chest when the chest is not handled: say "deadweight". Understand "dead" or "weight" or "deadweight" as the chest. Before pulling the wooden chest: now the chest is handled.

In the chest is a heavy dagger. The description of the dagger is "Set with red jewels and of a wicked aspect."

Before going down from the Field when the player is not stuck to something anchored:
    say "You don't quite dare simply leap into the darkness without some anchor." instead.

Before going down from the Field:
    let anchor be a random anchored thing which is stuck to the player;
    say "You lower yourself gingerly, hoping that [the anchor] holds your weight..."

Before going up from the Chasm:
    if the player cannot touch a rope which is stuck to an anchored thing which is in the Field, say "And how, precisely, do you mean to do that?" instead.

The Castle Hall is north of the Field. "All is desolate: the great hall has no roof, nor is there any glass in the windows. A staircase without banister ascends inside the wall to a musician's gallery without song."

The Musician's Gallery is above the Castle Hall. "Of its former cheery aspect only this remains to the Gallery: that chevrons of red and yellow are painted on the wall. But as these are streaked with rain and grime, the banister pulled away, the roof open to the sky, and the corners made a nesting place for birds, the consolation thereby afforded is but slight."

The pointed door is north of the Musician's Gallery and south of the Sinister Attic. It is a closed locked openable door. "A pointed door of particularly grim and uninviting aspect leads north." The pointed door is lockable and unevenly shaped. The description of the pointed door is "A door coming to a gothic point and fitted with iron fittings of great strength. The handle looks particularly well-attached." The iron key unlocks the pointed door.

A rule for reaching inside the Musician's Gallery:
    allow access.

A rule for reaching inside the Sinister Attic:
    allow access.

Instead of opening the pointed door for the first time:
    say "When you rattle at the door, there arises from beyond a terrible shrill noise as though something beyond exults in its imminent release."

After opening the trapped pointed door when the player

can see the pointed door:
    say "Thousands of bats fly from the pointed door, attacking you!";
    end the story.

After opening the pointed door when the player cannot see the pointed door: now the pointed door is untrapped; continue the action.

The pointed door can be trapped or untrapped. The pointed door is trapped.

Before pulling the pointed door: try opening the pointed door instead.

The player is unevenly shaped.

After going to the sinister attic:
    say "You have arrived at the goal of your quest!";
    end the story finally.

Test me with "x rope / pull rope / get chest / untie rope from chest / tie rope to me / down / get key / up / untie rope from stump / north / up / unlock pointed door with key / open it / tie rope to door / down / pull rope / up / north".

Test death with "x rope / pull rope / get chest / untie rope from chest / tie rope to me / down / get key / up / untie rope from stump / north / up / unlock pointed door with key / open it / g"

A string which can be cut into arbitrary lengths, and then tied back together.

"Snip!"

Length is a kind of value. 30 inch specifies a length. 20 in specifies a length. 50 inches specifies a length.

A string is a kind of thing. A string has a length. The length of a string is usually 36 inches.

Before printing the name of a string, say "[length] piece of ". Rule for printing the plural name of a string: say "[length] pieces of string".

Understand the command "cut" as something new. Understand "cut [length] from/off [something]" as trimming it by (with nouns reversed). Understand "cut [something] by [length]" as trimming it by. Understand the command "trim" as "cut".

Trimming it by is an action applying to one thing and one length.

Check trimming it by:
    if the length understood is 0 inches, say "You're approaching Zeno's string at this point." instead;
    if the length understood is greater than the length of the noun, say "[The noun] is only [length of the noun] long to start with." instead;
    if the length understood is the length of the noun, say

```
"[The noun] is already exactly [length of the noun] long."
instead.


Carry out trimming it by:
    now the length of the noun is the length of the noun
minus the length understood;
    let the other half be a random string in the string
repository;
    now the length of the other half is the length
understood;
    move the other half to the player.


Report trimming it by:
    reset string lengths; [we will define this in a moment;
it helps guarantee that our descriptions of the strings are
correct when we write the output list]
    say "You now have [a list of strings carried by the
player]."


Understand "cut [something] in half" as halving. Halving
is an action applying to one thing.


Carry out halving:
    let half measure be the length of the noun divided by
2;
    now the length understood is half measure;
    try trimming the noun by half measure.
```

This fudges slightly, since an odd-length string will be divided into uneven halves. Keeping track of fractional inches would complicate matters, though, so let's assume for now that this doesn't matter.

```
The player carries a string.


The Scissors Room is a room.
```

```
The string repository contains 35 strings.
```

Since our initial string is 36 inches long and it is impossible for the player to divide it into pieces smaller than an inch each, we need a total of 36 items to represent all the string-bits: one that the player carries at the outset, and 35 others. We should bear in mind that it is usually a good idea to use the smallest number of spare objects we can get away with: writing a game that required 1000 strings in the string repository would place silly demands on the resources of the system, so it's best to avoid that sort of thing if possible.

Now with a bit of fiddling we can also teach Inform to recognize descriptors such as "the shortest string":

```
Ordinariness is a kind of value. The ordinarinesses are
longest, medium, shortest. A string has an ordinariness.
Understand the ordinariness property as referring to a
string.


Definition: a string is small if its length is 2 in or less.
Definition: a string is large if its length is 20 in or more.


Before reading a command:
    reset string lengths.


To reset string lengths:
    let upper measure be the length of the largest visible
string;
    let lower measure be the length of the smallest visible
string;
    repeat with item running through strings:
        now the ordinariness of the item is medium;
        if the length of the item is the upper measure, now
the item is longest;
        if the length of the item is the lower measure, now
the item is shortest.
```

```
After reading a command:
    if the player's command includes "shorter", replace
the matched text with "shortest";
    if the player's command includes "longer", replace the
matched text with "longest".
```

```
Instead of tying a string to a string:
    move the second noun to the string repository;
    now the length of the noun is the length of the noun
plus the length of the second noun;
    decrease the length of the noun by 1 inch;
    say "You end up with [a noun], as some is taken up
by the knot."
```

This is still a little incomplete because we cannot refer to strings by their lengths, as in "the 2 inch string" and so on. To do this, we borrow a line from the chapter on Understanding:

```
Understand the length property as referring to a string.
```

```
Test me with "trim string by 4 in / cut longer string in half
/ cut longest string in half / cut shortest string in half / g /
g / tie longest string to shortest string / tie longest string
to medium string / i / x 16 inch string / drop 8 inch string
/ i".
```

Example **Rules of Attraction** *WI*

A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

Often we have some salient features of an object that we want to make sure the player notices whenever looking at the item in a room or in inventory. At other times, we may prefer to allow the name of the item to be printed bare. So for instance:

```
"Rules of Attraction"
```

A metal form is a kind of thing. A magnet is a kind of metal form.

Every turn:
    repeat with item running through nonmagnetic metal forms which are not part of something:
        if item is in a container which contains a magnet (called attractor):
            say "[The item] sticks to [the attractor].";
            now the item is part of the attractor.

The horseshoe magnet is a magnet carried by the player. The nail is a metal form carried by the player. The Barn is a room. In the Barn is a bucket. In the bucket is a metal form called the iron hook.

Definition: a thing is nonmagnetic if it is not a magnet.

Rule for printing room description details of a magnet (called attractor): if something is part of the attractor, say " (stuck to which [is-are the list of things which are part of the attractor])".

After printing the name of a magnet (called attractor) while taking inventory:
    if something is part of the attractor, say " (stuck to which [is-are the list of things which are part of the attractor])".

Before taking a touchable thing which is part of a magnet (called attractor):
    move the noun to the holder of the attractor.

Test me with "i / put horseshoe in bucket / look / get horseshoe / i / drop horseshoe / i / look / get all / put all in bucket / i / x magnet / get nail / i".

ExampleElectrified *WI*

Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances.

Suppose we want to prevent the player from touching anything electrified -- not just as a response to TOUCH OBJECT, but at any time when the action would require contact with the object in question.

"Electrified"

A thing can be safe or electrified. A thing is usually safe.

The Open Field is a room. "At this end of the field is a wire fence separating farm country from the government testing grounds beyond." The wire fence is an electrified thing in Open Field. It is scenery. The description of the wire fence is "Built into the fence is [a list of things which are part of the fence]." The scary box is an electrified container. It is part of wire fence. In the scary box is an alluring prize.

The player carries a flashlight, a grappling hook, a very thick rubber glove, and a length of rope. The glove is wearable.

This is the electrocution-wisdom rule:
    if the player wears the very thick rubber glove, make no decision;
    if the action requires a touchable noun and the noun is electrified, say "You fear touching [the noun]."

instead;
    if the action requires a touchable second noun and the second noun is electrified, say "You fear touching [the second noun]." instead.

The electrocution-wisdom rule is listed before the basic accessibility rule in the action-processing rules.

Before touching the scary box:
    say "You can't help noticing a bright red sticker on the surface of the box." [This rule will fire even if we are not wearing the glove, because Before rules occur before basic accessibility.]

Instead of opening the scary box:
    say "The scary box seems to be super-glued shut." [This one won't, because Instead rules occur after basic accessibility.]

Test me with "touch fence / touch box / open box / wear glove / open box".

ExampleWitnessed 1

A kind of battery which can be put into different devices, and which will lose power after extended use.

The following example makes fairly ample use of material that we haven't seen yet, but gives some idea of the flexibility of every turn rules. Suppose we want to have a number of electrical devices, all of which may be powered by a set of batteries. The batteries will all need to be discharged as they are used (regardless of what device they happen to be controlling at the moment). So:

"Witnessed"

A battery is a kind of thing. A battery has a number called charge. The charge of a battery is usually 15.

Every turn:
    repeat with hollow running through battery compartments:
        if the hollow is part of a switched on device (called the machine):
            if a battery (called cell) is in the hollow:
                decrement the charge of the cell;
                carry out the warning about failure activity with the machine;
                if the cell is discharged, carry out the putting out activity with the machine;
            otherwise:
                carry out the putting out activity with the machine.

Warning about failure of something is an activity.

Rule for warning about failure of a device (called the machine):
    if a random battery compartment which is part of the machine contains a battery (called the power source):
        if the charge of the power source is 2, say "[The machine] is obviously going to go out quite soon."

Putting out something is an activity.

Rule for putting out a device (called the machine):
        say "[The machine] loses power and switches off! [line break]";
        silently try switching off the machine.

```
A battery compartment is a kind of container. A battery
compartment is usually closed and openable. One
battery compartment is part of every device. Instead of
inserting something which is not a battery into a battery
compartment, say "Only batteries should go in a battery
compartment."
```

And to get rid of annoying messages like "Which would you like to close, the flashlight or the flashlight's battery compartment?" when only the compartment is closable, we might add some understanding instructions:

```
Understand "turn on [device]" as switching on.

Understand "turn off [device]" as switching off.

Understand "open [openable closed thing]" as opening.

Understand "close [openable open thing]" as closing.

Understand "put [something] in [container]" as inserting
it into.

Instead of opening a device, try opening a random
battery compartment which is part of the noun. Instead
of closing a device, try closing a random battery
compartment which is part of the noun. Instead of
inserting a battery into a device, try inserting the noun
into a random battery compartment which is part of the
second noun.

Instead of switching on an empty device:
    say "Nothing happens, perhaps because there isn't a
charged battery in [the noun]."
```

Instead of switching on a battery compartment which is part of a device (called the power user), try switching on the power user.

Definition: a device is empty:
    if a random battery compartment which is part of it contains a battery (called the power source):
        if the power source is discharged, yes;
        no;
    yes.

Definition: a battery is discharged if its charge < 1.

A light source is a kind of device. Carry out switching on a light source: now the noun is lit. Carry out switching off a light source: now the noun is unlit.

The flashlight is a light source. A D battery is a battery carried by the player.

The cassette recorder is a device. Every turn: if the cassette recorder is switched on, say "The cassette recorder hisses faintly."

Rule for warning about failure of the cassette recorder:
    if a random battery compartment which is part of the cassette recorder contains a battery (called the power source):
        if the charge of the power source is 2, say "The hiss from [the cassette recorder] begins to warble ominously."

The player wears a backpack. The backpack is openable. In the backpack is the flashlight and the cassette recorder.

The description of the cassette recorder is "Useful both for recording your notes and for capturing any odd ghostly sounds you may hear."

The description of the backpack is "An old familiar pack, which you know so well that you can find all its pockets and take things in and out of it in pitch darkness. To avoid it showing up oddly in photographs, it is entirely black, with no shiny or metallic attachments."

The description of the flashlight is "You bought a new one just for this occasion, because you were worried about bringing something too small or light. This is a heavy-duty flashlight with an adjustable-focus beam. The case is made of metal, rather than plastic, and there is a spare light-bulb inside as well. You've put a band of masking tape around the handle and written in your initials in red marker.

There is a piece of red cellophane attached to the business end of the flashlight to keep it from being overly bright."

The red cellophane is part of the flashlight.

Instead of doing something to the red cellophane: say "You need the cellophane on the flashlight so that using it does not completely destroy your night vision."

Thirtieth Street Station is a room. "A huge, high,

rectangular room with coffered ceilings, which looks grand but mostly makes you feel lonely and small. There are long benches in rows down the middle of the room, and an information desk with the train times, and a series of ticket windows, none of which matters very much at the moment."

The benches are an enterable supporter. They are scenery in the Station. The information desk is scenery in the Station. Some ticket windows are scenery in the Station. Instead of examining scenery in the Station: say "You're fairly sure that whatever is going on here has nothing to do with [the noun]." Understand "window" as ticket windows.

The mural is fixed in place in Thirtieth Street. "At the north side of the station is a particularly pointless and empty annex to the main room. It is dominated by a huge relief of sorts, and this is what you remember." Understand "metal" or "relief" or "huge" as the mural. The description of the mural is "It is both stylized and confusing, but you think it might be supposed to represent the various tasks and occupations of Philadelphia's population. The portions closer to the ground look as though they have recently been subjected to a light dusting of talcum powder. No unusual prints are evident."

The wind chimes are fixed in place in Thirtieth Street. "Carefully attached to the wall with a piece of duct tape and a hook is a light-weight set of wind chimes. Someone else has been here before you, it seems." The description is "Several of your friends use wind chimes as a sort of ghost alarm, since ghosts sometimes cause very localized movements of air when there is no natural breeze."

And this last bit, borrowed from the chapter on Understanding, adds some special instructions to help Inform decide when the player is likely to be referring to a compartment and when he's likely to be referring to the device itself.

```
Does the player mean doing something other than
searching to a battery compartment: it is unlikely. [We
discourage Inform from choosing a compartment when
the player uses just the name of a device or the word
'battery'.]
```

We also need to deal with commands like PUT BATTERY IN FLASHLIGHT, where Inform might construe BATTERY as the D battery, the flashlight's battery compartment, or the cassette recorder's battery compartment -- and might also construe FLASHLIGHT as either the flashlight's battery compartment or the flashlight itself.

```
Does the player mean inserting into a battery
compartment:
    if the noun is nothing:
        it is very likely;
    otherwise:
        make no decision.
```

```
Does the player mean inserting a battery compartment
into: it is very unlikely.
```

```
Does the player mean inserting something into a device:
it is unlikely.
```

```
Does the player mean searching a battery compartment:
it is very likely.
```

```
Test me with "test first / test second".
```

```
Test first with "i / open flashlight compartment / put
battery in it / turn on flashlight / take d battery / open
cassette compartment / turn on cassette / put battery in
cassette compartment / turn on cassette / z / z / z / z".
```

Test second with "get d battery / put d battery in flashlight compartment / turn on flashlight / z / z / z / z / z / z / turn off flashlight / z / z / turn on flashlight / z".

Example Thirst 2

A campfire added to the camp site, which can be lit using tinder.

"Thirst"

The player carries a waterskin. The waterskin can be full, partly drained, or empty. The waterskin is full. Understand "water" as the waterskin.

Instead of drinking the waterskin when the waterskin is empty:
    say "There is no water left."

Instead of drinking the waterskin: if the waterskin is partly drained, now the waterskin is empty; if the waterskin is full, now the waterskin is partly drained; say "You drink a long draught."

After printing the name of the waterskin: say " ([waterskin condition])"

Campsite is a room. "It is solid night now, and the stars have come out. Unfamiliar stars. On the other side of the valley -- a valley round-bottomed but shallow, like a soup bowl -- burn other campfires, most likely bandits. Their voices do not carry, but the smoke rises and obscures the starlight over that way."

A sleepsack is an enterable container in the Campsite. "Your sleepsack is laid out in a pocket of sandy soil and coarse grass."

The sandy soil, the stars, the distant campfires, and the coarse grass are scenery in the Campsite. Understand "smoke" as the campfires. Instead of listening in the presence of your campfire: say "All you hear are the reassuring snaps and cracks of the sticks in your fire." Understand "campfires" or "fires" as the distant campfires.

Your campfire is scenery in the Campsite. Instead of pushing, pulling, turning, tasting, or touching your campfire, say "You would burn yourself." Understand "fire" as your campfire. The description of your campfire is "A reassuring protection against wild animals and cold."

The description of the stars is "You invent constellations for them. The slingshot. The scroll. The heart (upside down)."

Instead of going nowhere when the player is in Campsite:
    say "Now is not the time for wandering, alone in the dark. Better to keep here[if your campfire is visible], by the fire[end if]."

Singing is an action applying to nothing. Understand "sing" as singing.

Instead of singing:
    say "You sing, deep and low, a song from home. It is

a good night for singing and the song raises your spirits."

The player carries a tinderbox. The tinderbox contains a flint, a steel, some tinder, and a patch of carbonized cloth. The description of the flint is "A flat grey stone with flaked edges." The description of the steel is "Curved so that you can hold it over the knuckles of your right hand and strike it against the flint. There is a knack to it. Those without the knack end up with bloody knuckles and no fire." The steel is wearable. The description of the tinder is "Dried grass and similar." The description of the cloth is "The little, precious, spark-preserving scraps without which the fire would never begin."

Instead of attacking the flint when the steel is not worn by the player:
    say "You must wear the steel over your knuckles, in order to hit the flint at the best angle."

Instead of attacking the flint when the steel is worn and the cloth is not in the location:
    say "Though you strike the flint sharply with the steel and throw sparks, they have nothing to catch on, since the patch of cloth does not lie beneath."

Instead of attacking the flint when the steel is worn and the cloth is in the location and the cloth is not unlit:
    say "The patch of cloth has already caught."

Instead of attacking the flint:
    now the cloth is glowing;
    say "You strike the flint against the steel and throw sparks onto the patch of cloth; they make tiny circles of orange there, which will only prosper if blown into flame."

Realistically, we ought to attach a randomization to this so that each step

of the fire-starting has a good chance of failure. But because our player may not be as patient as someone who actually needs a fire started, we allow him to succeed the first time in every case.

```
Ignition is a kind of value. The ignitions are whole,
fading, glowing, flaming. A thing has an ignition.


Blowing on is an action applying to one thing.
Understand "breathe on [something]" or "blow on
[something]" as blowing on.


Instead of blowing on the whole cloth:
    say "There is no point, since no sparks have caught
there."


Instead of blowing on the fading cloth:
    now the cloth is flaming;
    say "You blow on the faint sparks on the cloth and
turn them into the beginnings of flame."


Instead of burning the whole tinder:
    if the cloth is not flaming:
        say "The patch of cloth must catch flame before
you can light anything with it.";
    otherwise:
        now the tinder is flaming;
        now the cloth is nowhere;
        say "You light the tinder with the patch of cloth, and
have the elements of a fire."


Every turn:
    unless the cloth is flaming or the cloth is whole:
        now the ignition of the cloth is the ignition before
the ignition of the cloth;
        say "Now the patch of cloth is [ignition of the
cloth]."
```

Some kindling is in the campsite.

Instead of burning the whole kindling:
    if the tinder is not flaming:
        say "You need the tinder to be flaming, first.";
    otherwise:
        now the tinder is nowhere;
        now the kindling is nowhere;
        move the campfire to the location;
        say "You succeed in lighting yourself a proper campfire.";
        now the printed name of Campsite is "By The Campfire".

Test me with "i / drink water / i / drink water / i / wear steel / get flint / get cloth / drop cloth / get tinder / hit flint / blow on cloth / burn tinder / burn kindling / look".

Example **Bruneseau's Journey** *WI*

A candle which reacts to lighting and blowing actions differently depending on whether it has already been lit once.

"Sire," said the Minister of the Interior to Napoleon, "yesterday I saw the most intrepid man in your Empire." - "What man is that?" said the Emperor brusquely, "and what has he done?" - "He wants to do something, Sire." - "What is it?" - "To visit the sewers of Paris."

This man existed and his name was Bruneseau.

- Victor Hugo, Les Miserables

Let's say that our intrepid explorer has a candle that can be lit and blown out again, and should accordingly appear unlit, burning, or partly burnt:

"Bruneseau's Journey"

The Sewer Beneath St Denis is a room. "A narrow, stone-lined passageway, with only a little ledge to walk above the level of the refuse that flows down towards the Seine."

The candle is carried by the player. The description of the candle is "A candle, [if the candle has been lit]partially burnt[otherwise]still in pristine condition with untouched wick[end if]."

Instead of examining the lit candle, say "It burns with a pure heart."

The block burning rule is not listed in the check burning rules.

Instead of burning the lit candle:
    say "The candle is already lit."

Check burning:
    if the noun is not the candle, say "[The noun] cannot profitably be set on fire."

Carry out burning the candle:
    now the candle is lit.

```
Report burning:
    if the candle had been lit, say "You relight the
candle.";
    otherwise say "You light the candle for the first time.".



Understand "blow out [something]" as blowing out.
Understand "blow [something] out" as blowing out.
Blowing out is an action applying to one thing.



Carry out blowing out the candle:
    now the candle is unlit.



Report blowing out:
    if the noun is the candle and the candle was lit, say
"You blow out [the noun].";
    otherwise say "You blow on [the noun], to little effect."



Test me with "x candle / light candle / x candle / blow
out candle / x candle".
```

We must be careful: "if the noun was lit" would throw errors because past-tense rules can only be applied to specific items, not to variables that could be anything.

Example **The Cow Exonerated** *WI*

Creating a class of matches that burn for a time and then go out, with elegant reporting when several matches go out at once.

Here we create a class of matches that can be used to burn other objects. Our objectives are as follow:

Burned objects other than matches should be removed from play instantly (just as edible objects are instantly eaten). We could give everything its own burning duration, but that complicates matters and

allows for fire to spread from one object to another; for an example of how to do that, see the example "In Fire or in Flood".

Matches should be described to show whether they are burning or extinguished, and when the parser chooses one of several identical matches, it should make very clear which match it has selected.

The game must sensibly select and, if necessary, automatically light new matches to carry out a >BURN THING command.

The matches must burn for a set number of turns before going out, never to be used again.

And finally, the part for which the text will be useful: when several matches go out in the same turn, we want the game to print

```
Four matches go out.
```

rather than

```
A match goes out.
A match goes out.
A match goes out.
A match goes out.
```

This last function appears down in Section 3, if we wish to skip ahead and look at it.

```
"The Cow Exonerated"


Section 1 - Simple Burning


Understand the commands "light" and "burn" as
something new.


Understand "burn [something] with [strikable-match]" as
burning it with. Understand "burn [something] with
```

[something preferably held]" as burning it with. Burning it with is an action applying to one thing and one carried thing.

Understand the command "light" as "burn".

A thing can be flammable or impervious. A thing is usually impervious.

Check burning something with something (this is the burn only with flaming matches rule):
    if the second noun is not a strikable-match, say "You can only light things with matches." instead;
    if the second noun is not flaming, say "[The second noun] needs to be burning first." instead.

Check burning something with something (this is the burn only flammable things rule):
    if the noun is impervious, say "[The noun] cannot be burned." instead.

Check burning something with something (this is the burn only things not held rule):
    say "[one of]It occurs to you to set down [the noun] before burning, just for safety's sake. [or]Again, you decide to put down [the noun] prior to burning. [or]You try setting down [the noun] as usual. [stopping][run paragraph on]";
    silently try the player dropping the noun;
    if the player encloses the noun, stop the action.

Carry out burning something with something (this is the simplistic burning rule):
    now the noun is nowhere.

```
Report burning something with something:
    say "You burn up [the noun]."


Rule for implicitly taking the second noun while burning
something with something which is not a strikable-
match:
    say "You can only light things with matches.";
    stop the action.
```

Section 2 - Matches

The word "matches" is used by Inform to compare snippets of text (see "Reading a command" in the Activities chapter). This can sometimes cause awkwardness if we also have a kind called "match", so for the occasion we will give our matches a more specialized name, never visible to the player:

```
A strikable-match is a kind of thing. The plural of
strikable-match is s-matches.


A strikable-match has a number called duration. The
duration of a strikable-match is usually 3.


Rule for printing the name of a strikable-match: say
"match".
Rule for printing the plural name of a strikable-match:
say "matches".


Understand "match" as a strikable-match. Understand
"matches" as a strikable-match.


Flame-state is a kind of value. The flame-states are
burnt, flaming, and new. Understand "burning" or "lit" as
flaming. Understand "unused" as new.
```

A strikable-match has a flame-state. A strikable-match is usually new. Understand the flame-state property as describing a strikable-match.

Before printing the name of a strikable-match while asking which do you mean:
    say "[flame-state] ".

Before printing the name of a strikable-match while taking inventory:
    say "[flame-state] ".
Before printing the plural name of a strikable-match while taking inventory:
    say "[flame-state] ".

Before printing the name of a strikable-match while clarifying the parser's choice of something:
    if not taking inventory, say "[flame-state] ".

After printing the name of a strikable-match (called special-target) while clarifying the parser's choice of something:
    if the player carries the special-target:
        say " you're carrying";
    otherwise if the special-target is in the location:
        say " on the ground";
    otherwise:
        say " [if the holder of the special-target is a container]in[otherwise]on[end if] [the holder of the special-target]".

Understand "strike [something]" as attacking.

Understand "strike [strikable-match]" as striking. Striking

is an action applying to one carried thing.

Understand "burn [strikable-match]" as striking.

Does the player mean striking a new strikable-match:
    it is very likely.

Does the player mean striking a burnt strikable-match:
    it is unlikely.

Check striking a strikable-match (this is the strike only new matches rule):
    if the noun is burnt, say "[The noun] has already burnt down and cannot be relit." instead;
    if the noun is flaming, say "[The noun] is already burning." instead.

Carry out striking a strikable-match (this is the standard striking rule):
    now the noun is flaming;
    now the noun is lit.

Report striking a strikable-match (this is the standard report striking rule):
    say "You light [the noun]."

Before burning something with a new strikable-match (this is the prior lighting rule):
    say "(first [if the player does not carry the second noun]taking and [end if]lighting [the second noun]) [command clarification break]";
    silently try striking the second noun;
    if the second noun is not flaming, stop the action.

```
Rule for implicitly taking a strikable-match (called target)
while striking:
    try silently taking the target.



Does the player mean burning something with a flaming
strikable-match:
    it is very likely.



Does the player mean burning something with a new
strikable-match:
    it is likely.



Does the player mean burning something with a burnt
strikable-match:
    it is unlikely.



Instead of burning a burnt strikable-match with
something:
    say "[The noun] is completely consumed and cannot
be relit."



Section 3 - Putting the Matches Out



Every turn:
    let N be 0; [here we track how many matches are
being put out during this turn, so that we don't have to
mention each match individually if several go out during
the same move]
    repeat with item running through flaming s-matches:
        decrement the duration of the item;
        if the duration of the item is 0:
            now the item is burnt;
            now the item is unlit;
            if the item is visible, increment N;
```

```
    if N is 1:
        say "[if the number of visible flaming s-matches is
greater than 0]One of the matches [otherwise if the
number of burnt visible s-matches is greater than 1]Your
last burning match [otherwise]The match [end if]goes
out.";
    otherwise if N is greater than 1:
        let enumeration be "[N in words]";
        if N is the number of visible s-matches:
            if N is two, say "Both";
            otherwise say "All [enumeration]";
        otherwise:
            say "[enumeration in title case]";
        say " matches go out[if a visible strikable-match is
flaming], leaving [number of visible flaming s-matches in
words] still lit[end if]."
```

Section 4 - Scenario

```
Old Chicago is a room.
```

```
The player carries a flammable thing called a log.
Understand "wooden" and "wood" as the log.
```

```
The player carries two s-matches. The matchbox is an
open openable container. It contains five s-matches.
The player carries the matchbox.
```

```
When play begins:
    now every strikable-match carried by the player is
flaming;
    now every strikable-match carried by the player is lit.
```

```
Test me with "i / burn match / i / i / burn log with match /
burn matchbox with match / i".
```

Example In Fire or in Flood *WI*

A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

"In Fire or in Flood"

Part I - Rules for combustion

Heat is a kind of value. The heats are whole, damp, and flaming. A thing has a heat. A thing is usually whole.

A thing has a number called endurance. The endurance of a thing is usually 5. A thing has a number called turns of burning. A thing can be flammable or flame-retardant.

Before printing the name of something flaming:
    say "flaming ".

Before burning something when the player is not carrying something flaming:
    if a flaming portable thing (called the lighter) is touchable:
        say "(with [the lighter], which you first take) [command clarification break]";
        try taking the lighter.

Instead of burning something when the player is not carrying something flaming:
    say "You would first need a fire source."

```
Instead of burning something flame-retardant:
    say "[The noun] is not the sort of thing that catches
fire."


Instead of burning something flammable when the
player is carrying something flaming (called the flame
source):
    say "You light [the noun] with [the flame source].";
    now the heat of the noun is flaming.


Instead of burning something when the player is in the
noun:
    say "That seems dangerous given that you yourself
are in [the noun]."


Instead of burning something when the player is on the
noun:
    say "That seems dangerous given that you yourself
are on [the noun]."


Instead of examining something:
    say "Hm, the [printed name] appears to be [heat]."


Before taking a flaming thing:
    let turns remaining be the endurance of the noun
minus the turns of burning of the noun;
    if turns remaining is less than two, say "There's no
portion of [the noun] sufficiently cool for you to pick up."
instead.
```

But that's only a small part of the battle. The thing about fire is that it keeps on doing fiery things even when the player is otherwise occupied: destroying items that are on fire, and spreading to other things nearby. So we need a set of rules for the fire's behavior.

```
Every turn when something is flaming:
    follow the fire rules.


The fire rules is a rulebook.


A fire rule (this is the can't hold flaming objects rule):
    repeat with item running through flaming things:
        if the item is held by the player:
            let turns remaining be the endurance of the item
minus the turns of burning of the item;
            if turns remaining is less than two:
                say "[The item] becomes too hot to hold! ";
                try dropping the item;
                if the item is held by the player, say "This is
certainly painful."


A fire rule (this is the flames spread rule):
    repeat with item running through flaming things:
        if the turns of burning of the item is one:
            spread the flames from the item.


A fire rule (this is the fire destroys things rule):
    now started printing is false;
    repeat with item running through flaming things:
        increment the turns of burning of the item;
        if the turns of burning of the item is greater than the
endurance of the item, destroy the item;
    if started printing is true, say "[paragraph break]";
    now started printing is false.
```

Because we've labelled all the fire rules, we could swap their order, or turn some of them off, while allowing the others run as usual. For instance, if there were a pair of fireproof gloves in the game, we might want to turn off the "can't hold flaming objects rule" whenever the player is wearing them.

This sort of flexibility is especially useful in the context of extensions. Someone writing an extension about burning would have no way of anticipating the need for a Fireproof Gauntlet of Thog, but the author would nonetheless be able to implement one easily.

```
Definition: a thing is vulnerable if it is flammable and it is
whole.
```

The contact between things is a critical factor when it comes to fire, so we might add a couple of conditional relations do determine what is touching what.

```
Reliance relates a thing (called X) to a thing (called Y)
when X is part of Y or X is in Y or X is on Y. The verb to
be relying on means the reliance relation.
```

```
Contact relates a thing (called X) to a thing (called Y)
when X is relying on Y or Y is relying on X. The verb to
be joined to means the contact relation.
```

Having these at our disposal makes it much tidier to write what happens next:

```
To spread the flames from (item - a thing):
    now started printing is false;
    if the item is joined to a flammable whole thing (called
the sacrifice):
        if the sacrifice is visible:
            now started printing is true;
            say "Flames engulf [the list of flammable whole
things which are joined to the item].";
        now all the flammable whole things joined to the
item are flaming.
```

```
Started printing is a truth state that varies. Started
printing is false.
```

```
To destroy (item - a thing):
    let home be the holder of the item;
    if the item is part of something (called the
superstructure), let home be the holder of the
superstructure;
```

```
    if the item is visible:
        now started printing is true;
        say "[The item] burns away[if something is relying
on the item], leaving [a list of things which are relying on
the item] behind[end if]. ";
    if something is relying on the item,
        now all the things which are relying on the item are
in the home;
    now the item is nowhere;
    now the item is damp;
    now every flaming thing which is part of the item is
damp.



To destroy (item - a door):
    let home be the holder of the item;
    if item is visible:
        now started printing is true;
        say "[The item] burns away[if something flame-
retardant is part of the item], leaving [a list of flame-
retardant parts of the item] behind[end if]. ";
    if home is a room, now all of the flame-retardant parts
of the item are in the home;
    now the item is damp;
    now the item is open;
    now the item is unopenable.



Before printing the name of a damp door:
    say "burnt-out frame of ".



Instead of opening or closing a damp door:
    say "[The noun] can no longer be opened or closed in
any meaningful sense."



Instead of doing something other than examining or
dropping to a flaming thing when the turns of burning of
the noun is greater than 1:
    say "Fire has too thoroughly engulfed [the noun] for
that to be a good idea."
```

```
Instead of taking something when the noun is in a
flaming thing (called the receptacle):
    say "You don't quite dare reach into [the receptacle]."

Instead of touching something which is within a flaming
thing (called the receptacle):
    say "It seems a little risky since [the receptacle] is on
fire."

Instead of turning something when the noun is
contained in a flaming thing (called the receptacle):
    say "It seems a little risky since [the receptacle] is on
fire."

Instead of pushing or pulling something when the noun
is inside a flaming thing (called the receptacle):
    say "[The receptacle] deters you."

Before burning something which is in a container when
the holder of the noun contains the player:
    say "This could make things toasty for you..."
```

And that completes the rules which cover burning: things can catch fire, fire will spread, and gradually consume the world in flames. All of that was general and could be used in any setting, but we now provide a small game to show it off.

```
Part II - Escape from the Library of the Dead

The Library of the Dead is a room. "This room -- little,
dank, stone -- is filling with some miasma you do not
quite dare breathe. It is imperative that you get out."

The desk is a flammable supporter in the Library. A
```

drawer is part of the desk. The drawer is a flammable closed container. It is openable, lockable, and locked. The desk is scenery.

A box is in the Library. A metal hinge is part of the box. The hinge is flame-retardant. The box is open, flammable, and openable. The shroud of Laertes is a flammable thing in the box.

Instead of examining something when something is part of the noun:
    say "You note [the list of things which are part of the noun]."

The world's last manuscript of the Psychagogoi by Aeschylus is on the desk. The manuscript is flammable. The manuscript has endurance 1.

The torch is a flammable flaming thing carried by the player. It has endurance 60. The asbestos sack is a flame-retardant player's holdall in the drawer.

The trapdoor is up of the Library and east of the Plaza. The trapdoor is a door. It is flammable, closed, lockable, and locked. "A trapdoor in the ceiling is your only hope of escape[if flaming]. Fortunately, it is rapidly burning through[end if]." The trapdoor has endurance 15.

Instead of going through the closed trapdoor, say "[The trapdoor] is closed."

We can then add a special fire rule to handle the trapdoor, which will be called as part of the same sequence. Again, this would be most important if the fire rules were part of a standard extension, and the trapdoor fire rule the author's own addition.

```
A fire rule:
    if the trapdoor is flaming and a random chance of 1 in
3 succeeds:
        let the caught thing be a random flammable whole
thing which can be touched by the trapdoor;
        if the caught thing is a thing:
            say "A spark from [the trapdoor] catches [the
caught thing]!";
            now the caught thing is flaming.



Instead of going to the Plaza:
    say "Out at last!";
    end the story finally.



Test me with "get manuscript / get shroud / light desk /
look / g / open drawer / look / g / g / g / get sack / put
shroud in sack / put manuscript in sack / close sack /
light trapdoor / look / g / g / g / g / g / g / g / g / g /
g / g / g / up".
```

---

84 ExampleGrilling *WI*

A grill, from which the player is not allowed to take anything lest he burn himself.

Descriptions of objects can be used in "Instead" rules: we can not only say "Instead of taking the steak", but also "Instead of taking something" or "Instead of taking something which is on the grill".

That last rule is useful if, for example, we want to prevent the player from interacting with anything on a specific supporter:

```
"Grilling"



The Patio is a room. The Patio contains a grill and an
```

```
ice chest. The ice chest contains a cold beer. On the
grill are a steak and a hot dog.


Mom is a woman in the Patio.


Instead of taking something which is on the grill:
    say "'Hey, you'll burn yourself,' says Mom."


Test me with "get steak / get all from grill / get all".
```

We could just as easily adapt this rule to affect a container: "Instead of taking something which is in the ice chest," for example.

Note also that in older versions of Inform, the pattern "get all from..." was treated differently from "get steak", and had to be accounted for separately. This is no longer the case; this instead of taking... rule will handle all the phrasings which the player might use to try to acquire this object.

ExampleEntropy

All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

```
"Entropy"


Heat is a kind of value. The heats are frosty, cold, cool,
room temperature, warm, hot, and scalding. Everything
has a heat. The heat of a thing is usually room
temperature.


Every turn:
```

repeat with item running through things which are not in an insulated container:
if the heat of the item is greater than room temperature, now the heat of the item is the heat before the heat of the item;
if the heat of the item is less than room temperature, now the heat of the item is the heat after the heat of the item.

Definition: a container is insulated if it is closed and it is opaque.

The vacuum thermos is an opaque closed openable container carried by the player. In the vacuum thermos is a frosty thing called an ice cube.

Every turn:
if the heat of the ice cube is greater than cold:
if the ice cube is visible, say "The ice cube melts! 'HA ha,' says Maxwell, in a very unsporting, some might say demonic, way.";
now the ice cube is nowhere.

Before printing the name of something: say "[heat] ".

Equilibrium is a room. "A perfectly smooth chamber sealed from the outside world. You can't at this moment work out where the exit is, though possibly that is just because the lighting is so very very even and diffuse. And doesn't come from anywhere that you can see, either."

Maxwell is a man in Equilibrium. "Maxwell perches awkwardly on a stool across from you[if Maxwell has something], holding [a list of things carried by Maxwell] [end if]." He is carrying a box of Chinese food. The

Chinese food is scalding. "A discarded [item described] lies on the floor." The description of Maxwell is "He has the faintly peevish look of one who has not been properly fed."

Every turn when Maxwell has the food:
    if the heat of the Chinese food is greater than warm, say "Maxwell takes a bite, and swears.";
    if the heat of the Chinese food is warm, say "Maxwell eats as fast as he can, enjoying the food while it's at just the right temperature.";
    if the heat of the Chinese food is less than warm:
        say "Maxwell sadly stabs at his leftovers with a chopstick, but does not try to eat any more.";
        move the food to the location.

Test me with "z / z / open thermos / close thermos / open thermos".

ExampleMasochism Deli

Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

Here the player has several potatoes; we would like to make him more likely to drop the hot one, and more likely to pick up the cold one, all else being equal. At the same time, we want to phrase our rules so that they don't make the player try to take something he's already holding, or drop something he isn't.

So:

"Masochism Deli"

The Masochism Deli is a room. "Recent restructurings of corporate policy restrict the 'lunch hour' to exactly thirty-two minutes, which means that no one has time to go out. Instead, you and your coworkers eat here, in the

company's very own themed lunch room."

The plural of potato is potatoes. A potato is a kind of thing. A potato is edible.

Temperature is a kind of value. The temperatures are hot and cold. A potato has a temperature. A potato is usually cold.

Understand the temperature property as describing a potato. Before printing the name of a potato (called subject): say "[temperature of subject] ". Before printing the plural name of a potato (called subject): say "[temperature of subject] ".

Does the player mean dropping a hot potato which is carried by the player: it is very likely.

Does the player mean taking a cold potato which is not carried by the player: it is very likely.

The player carries three potatoes.

After dropping a hot potato:
    say "The guy from Cube B sneers at your lack of potato-holding stamina."

When play begins: now a random potato is hot.

Test me with "inventory / drop potato / g / g / get potato / g / i / get potato".

---

331 **Example** **Hot Glass Looks Like Cold Glass** *WI*

Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

Suppose we have a situation where the player is allowed to talk about the heat of an object only if he's properly equipped to detect it.

"Hot Glass Looks Like Cold Glass"

Use scoring.

Heat is a kind of value. The heats are hot, warm, room temperature, and cold. A thing has a heat.

Understand the heat property as referring to a thing when the player wears the infrared goggles.

The Test Kitchen is a room. "Your own personal lab, ready for scrupulously scientific recipe research. You hope. The previous three runs of this did not go well." The pair of infrared goggles is carried by the player. The description is "A pair of head-mounted IR goggles which look very foolish when worn." The goggles are wearable.

A glass dish is a kind of container. A glass dish is transparent. Three room temperature glass dishes are on the counter. Two hot glass dishes are on the counter.

```
Two cold glass dishes are on the counter. The counter
is scenery in the Test Kitchen.


Instead of doing something other than examining to a
hot glass dish:
    say "Ow! Crikey! You swear, and Claudia makes a
sympathetic hiss. 'You're going to have a mark from that
for sure,' she comments.";
    decrease the score by 2.


Instead of doing something when a hot glass dish is the
second noun:
    say "You brush [the second noun], and wince, but
manage to conceal that from Claudia.";
    decrement the score.


Before printing the name of a glass dish when the player
wears the goggles: say "[heat] "


Before printing the plural name of a glass dish when the
player wears the goggles: say "[heat] "
```

So far, so good. Now, what if the player tries to GET HOT DISH when the goggles are off? "You can't see any such thing." doesn't seem like quite the right response: he can see such a thing. He just doesn't know which it is.

We could go on to write a mistake rule that would scold the player for trying "get [heat] [text]" when not wearing the goggles. The problem is that this would not cover any other phrasing of the command, nor would it account for all the many other things the player might try to do with an object specified by heat.

What we really want is to catch all instances of the player using the property name when not allowed to do so; and for this purpose we can borrow a trick from the chapter on Activities:

```
After reading a command:
    if the player wears the goggles, make no decision;
```

```
    if the player's command includes "[heat]":
        say "Without the IR goggles on, you cannot tell hot
things from cold at sight.";
        rule succeeds.
```

```
Claudia is a woman in the Test Kitchen. "Your assistant
Claudia stands by with [a list of things carried by
Claudia]." The description of Claudia is "Infinitely patient
and a very good stenographer. She is studiously
avoiding giving you any sort of look that might be
construed as mocking." Claudia carries a notepad, a
brined chicken breast, a blowtorch, and a cup of heavy
cream.
```

```
Instead of asking Claudia for something which is carried
by Claudia:
    move the second noun to the player;
    say "'Check, [second noun],' repeats Claudia, in the
tone of one who has seen too many medical dramas.
She does hand it over, though."
```

```
Instead of asking Claudia for something:
    say "She clears her throat faintly and glances at [the
second noun], as though to say that it's not hers to
give."
```

```
Test me with "get all / drop all / look / wear goggles /
look / ask claudia for cream / put cream in hot dish / put
cream in cold dish / remove goggles / get hot dish".
```

ExampleMagneto's Revenge

Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

"Magneto's Revenge"

The School for Gifted Youngsters is a room. Kitty Pryde is a woman in the School for Gifted Youngsters. "Kitty Pryde waits for you to say why you summoned her out of class." The description is "You see nothing special about Kitty Pryde. But that is only because she has learned to conceal her mutant powers: in fact she has the alternative name Shadowcat, and the ability to phase through solid objects."

The glass box is a container in the School for Gifted Youngsters. It is closed, openable, and transparent. In the box is a quantity of poisonous gas. In the box is a thing called the message from Magneto.

Instead of examining the message when the player is not carrying the message:
    say "You'd need to be holding it before you could read it."

Instead of opening the glass box: say "You're deterred by the swirling bottle-green mist within."

Check someone taking the gas (this is the gaseous object rule): rule fails.

Unsuccessful attempt by someone taking the gas: say "The gas isn't something one can pick up in one's bare hands."

Every turn:
    if the player can touch the gas:
        say "The gas has reached your lungs!";

```
end the story.


A rule for reaching inside something:
    if the person reaching is Kitty, allow access.


Persuasion rule for asking someone to try doing
something: persuasion succeeds.


Test me with "get message / kitty, get message / open
box / kitty, open box".
```

Now the player can get Kitty to take the message without releasing the poisonous gas:

```
Kitty Pride waits for you to tell her why you summoned
her out of class.


You can also see a glass box (closed) (in which are
some poisonous gas and a message from Magneto)
here.


>kitty, get message
Kitty Pryde picks up a message from Magneto.
```

ExampleInterrogation

A wand which, when waved, reveals the concealed items carried by people the player can see.

```
"Interrogation"
```

The X-Ray Vision Wand is carried by the player.

Instead of waving the X-Ray Vision Wand:
    say "Disappointingly, nothing happens."

Instead of waving the X-Ray Vision Wand when the player can see someone who is concealing something:
    say "The wand glows green. Immediately you see on the monitor [a list of things which are concealed by people who can be seen by the player]."

After printing the name of a thing (called target) which is carried by someone while waving the wand:
    say " (carried by [a random person who carries the target])"

The Interrogation Chamber is a room. "Despite its gothic name, this room is a rather civilized place for your work, with large plate-glass windows and a fitted carpet."

A thing can be secret or obvious. A thing is usually obvious.

Brian is in the Interrogation Chamber."Brian lounges against the wall." Brian carries a quantity of plastic explosive. The explosive is secret.

Janine is in the Interrogation Chamber. "Janine toys nervously with a laptop bag." Janine carries a chocolate biscuit, a laptop bag, and a microfilm. The microfilm is secret. The laptop bag is wearable. In the bag is a laptop computer.

Rule for deciding the concealed possessions of something: if the particular possession is secret, yes; otherwise no.

Instead of examining someone: say "[The noun] is openly carrying [a list of unconcealed things carried by the noun]."

Test me with "wave wand / examine janine / examine brian".

---

227 **ExampleTransmutations** *WI*

A machine that turns objects into other, similar objects.

Suppose we want to have a machine in our game that can transmute one item into another, similar object with different properties: a bag of jelly beans into a bag of jewels, for instance. Thus each item will be associated with some number of equivalents -- the other objects it can turn into. This is a handy use for group relations:

"Transmutations"

Workshop is a room.

Transmutation relates things to each other in groups.
The verb to become means the transmutation relation.

Definition: a thing is transmutable if it becomes more

than one thing. [* It always becomes itself.]

A thing can be valuable. Something valuable called a bag of jewels is carried by the player. It becomes the bag of gunpowder and the bag of jelly beans.

A thing can be dangerous. The bag of gunpowder is a dangerous thing.

The bag of jelly beans is an edible thing.

The machine is fixed in place in the workshop.

The can't insert into what's not a container rule does nothing when inserting something into the machine.

Check inserting something which is not transmutable into the machine:
    instead say "You can't transmute that."

To decide which thing is new form of (obj - edible thing): decide on a random valuable thing which becomes obj.

To decide which thing is new form of (obj - dangerous thing): decide on a random edible thing which becomes obj.

To decide which thing is new form of (obj - valuable thing): decide on a random dangerous thing which

```
becomes obj.


Carry out inserting something into the machine:
    now the noun is nowhere;
    now the player carries the new form of the noun.


Report inserting something edible into the machine:
    say "The machine clicks, whirrs, and spits out [a new
form of the noun]. You're rich!";
    rule succeeds.


Report inserting something dangerous into the machine:
    say "The machine clicks, whirrs, and in a shower of
flavor crystals, spits out [a new form of the noun].";
    rule succeeds.


Report inserting something valuable into the machine:
    say "The machine clicks, whirrs, and with a violent
roar, spits out [a new form of the noun].";
    rule succeeds.


    Test me with "i / put jewels in machine / i / put
gunpowder in machine / i / put beans in machine".
```

In this example we have only defined a single set of transmutable objects, but we could easily expand to include other groups.

(Thanks to Jesse McGrew for proposing this example.)

Example **Access All Areas**

The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

```
If somebody tries to walk through a closed door, the "can't go through
```

closed doors rule" usually stops them. This is a rule belonging to the "check going" rulebook. These names are fairly explanatory when written out, but hard to remember: fortunately we don't need to remember them, as the Index panel contains a full inventory of the check, carry out and report rules for every action, showing all of their names and the order in which they are checked. (We can also find out which rules are stopping an action by typing the testing command ACTIONS.)

Here we make the rule do nothing provided a condition holds:

```
"Access All Areas"

The extremely difficult door is north of the Standing
Room and south of the Room of Walking Upside Down.
It is a locked door.

The player is carrying the Pointy Hat of Liminal
Transgression. The hat is wearable.

The can't go through closed doors rule does nothing
when the Hat is worn.

Test me with "n / wear hat / n".
```

(The Pointy Hat may be useful in debugging a game, even if it never makes it into the final published work.)

This is the same case given above, but expanded just slightly to demonstrate that the names of the relations can also be printed, if we like:

```
"Number Study"

Abstraction is a room.

Parity relates a number (called N) to a number (called
M) when N minus M is even.

Joint magnitude relates a number (called N) to a
number (called M) when N plus M is greater than 7.

To chart (R - a relation of numbers):
    repeat with N running from 1 to 5:
        repeat with M running from 1 to 5:
            if R relates N to M, say "[N] <=> [M] by [R][line
break]";

When play begins:
    let L be { parity relation, joint magnitude relation };
    repeat with R running through L:
        chart R.
```

As this shows, we can even form lists of relations. The kind of L is "list of relations of numbers".

Example **The Fibonacci Sequence**

The modest Leonardo Fibonacci of Pisa will be only too happy to construct his sequence on request, using an array.

```
Fibonacci (a posthumous nickname) spread Arabic mathematical
learning across Europe in the 13th century, and it's curious that his name
lives on only for a single sequence.
```

```
"The Fibonacci Sequence"


Pisa is a room. Leonardo Fibonacci is a man in Pisa.
"The modest Italian mathematician, Leonardo Fibonacci
(1170-1250), beams at you."


Sequencing is an action applying to one number.
Understand "sequence [number]" as sequencing.


Instead of sequencing, say "You make a feeble attempt,
sketching in the sand, but it goes nowhere. Leonardo is
sympathetic. 'Often goes wrong for me, too, actually. I
didn't even invent the thing - the ancient Indians knew
about it first.'"


Persuasion rule for asking Leonardo to try sequencing:
persuasion succeeds.


Report Leonardo sequencing:
    let N be the number understood;
    say "Leonardo scratches his head and makes self-
deprecating remarks, before coming up with [the first N
terms of the Fibonacci sequence]."
```

An array need not be fixed in length, as the following example shows:

```
To decide what list of numbers is the first (F - a number)
terms of the Fibonacci sequence:
    let the Fibonacci sequence be {1, 1};
    let N be 3;
    while N < F:
        let the last term be entry (N - 1) of the Fibonacci
sequence;
        let the penultimate term be entry (N - 2) of the
```

```
Fibonacci sequence;
        let the next term be the last term plus the
penultimate term;
        add the next term to the Fibonacci sequence;
        increment N;
    decide on the Fibonacci sequence.



Test me with "sequence 20 / leonardo, sequence 20".
```

The result of "the first 20 terms of the Fibonacci sequence" is "1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584 and 4181". This is a sequence which has a knack of turning up in odd places - it was found in the 1970s to be related to the rings of florets in a sunflower, for instance - and here it is in a book about interactive fiction.

Example **Sieve of Eratosthenes**

The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

In the words of Wikipedia: "Eratosthenes of Cyrene (Greek Ερατοσθένης; 276 BC-194 BC) was a Greek mathematician, poet, athlete, geographer and astronomer." In the words of Tom Lehrer: "It's people like that who make you realise how little you've achieved in life."

A prime number is a number greater than 1 which is not a multiple of anything, so we can find the primes by starting with all the numbers and sieving out all the multiples of 2, then all the multiples of 3, and so on. Here we make our sieve of the unacceptable numbers (the "composite" or non-prime ones) first, then form a list of all the numbers, then sieve out the composites: what are left must be the primes.

```
"Sieve of Eratosthenes"



Alexandria is a room. Eratosthenes is a man in
Alexandria. "The haughty Greek mathematician,
Eratosthenes, glowers at you."
```

```
Sieving is an action applying to one number.
Understand "sieve [number]" as sieving.



Instead of sieving, say "You make a feeble attempt,
sketching in the sand, but it goes nowhere.
Eratosthenes smirks. 'I expect your friends call you
gamma, then?'"



Persuasion rule for asking Eratosthenes to try sieving:
persuasion succeeds.



Report Eratosthenes sieving:
    let N be the number understood;
    let the composites be a list of numbers;
    let I be 2;
    while I times I is at most N:
        if I is not listed in the composites:
            let J be I times 2;
            while J is at most N:
                add J to the composites, if absent;
                increase J by I;
        increment I;
    sort the composites;
    let the primes be a list of numbers;
    repeat with P running from 2 to N:
        add P to the primes;
    remove the composites from the primes;
    say "Eratosthenes sketches lines in the sand with the
air of much practice. 'The primes up to [N] are [the
primes]. The composites are [the composites].'"



Test me with "sieve 10 / eratosthenes, sieve 100".
```

While this could all be done more efficiently with an array, that's only because what we are sieving are numbers: sieving is a technique which can be used for non-numerical decisions, too.

Chapter 11: Out Of World Actions and Effects

Saving and Undoing; *§11.3. Helping and Hinting*; *§11.4. Scoring*; *§11.5. Settings and Status Checks Du*

Contents of *The Inform Recipe Book*
Chapter 10: Physics: Substances, Ropes, Energy and Weight
Chapter 12: Typography, Layout, and Multimedia Effects
Indexes of the examples

§11.1. Start-Up Features

When the story file starts up, it often prints a short introductory passage of text (the "overture") and then a heading describing itself, together with some version numbering (the "banner"). It is traditional that the banner must appear eventually (and one of the few requirements of the Inform licence is that the author acknowledge Inform somewhere, for which the banner is sufficient) but some designs call for a multi-turn prologue before the banner finally appears, and marks the start of play in earnest. **Bikini Atoll** demonstrates this.

If a story file represents the latest in a sequence of story files representing chapters in some larger narrative, it will need some way to pick up where its predecessor left off. This can be done with the aid of external files (in the Glulx format, at least). **Alien Invasion Part 23** shows how.

Another task we might want to perform a the beginning of play is to arrange any randomized features that are supposed to change from one playing to the next. We can add such instructions with "When play begins" rule, as in:

```
When play begins:
    now the priceless treasure is in a random room.
```

Since we may want to do something a bit more complicated than this, **Hatless** demonstrates effective and ineffective methods of distributing multiple objects (in this case, one randomly-selected hat per person).

See **Map** *for a way to generate a randomized maze at the start of play*

See **Food** *for a way to choose a random piece of candy to be poisonous*

See **Getting Acquainted** *for a way to choose a murderer from among the characters at the start of each story*

Start of Chapter 11: Out Of World Actions and Effects

Back to Chapter 10: Physics: Substances, Ropes, Energy and Weight: §10.11. Mathematics

Onward to §11.2. Saving and Undoing

Example 382: **Bikini Atoll** Delaying the banner for later.

Example 440: **Alien Invasion Part 23** Keeping a preference file that could be loaded by any game in a series.

Example 126: **Hatless** It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

---

§11.2. Saving and Undoing

A very few titles in the IF literature - very few being still too many, some would say - restrict the player's ability to save the story.

Removing the player's ability to UNDO is also a risky choice. Inform does provide the facility with the use option

Use undo prevention.

which makes it impossible to UNDO at any time (unless, that is, the player is playing on an interpreter that itself has a built-in UNDO feature -- these do exist). When it works, undo prevention safeguards a randomized story or combat session against

brute-force solutions, but it also means that the player who makes even a minor mistake of typing will be stuck with the undesired results.

In many cases it may be preferable to use some subtler method to enforce random effects in a story. Several extensions exist for Inform that either allow selective manipulation of the UNDO command or rig randomization to prevent UNDO and replay attempts.

Start of Chapter 11: Out Of World Actions and Effects

Back to §11.1. Start-Up Features

Onward to §11.3. Helping and Hinting

Example 210: **Spellbreaker** P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

Example 211: **A point for never saving the game** In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

---

§11.3. Helping and Hinting

IF is difficult to play: often harder than the writer ever suspects. Players are held up by what is "obvious", and they stumble into unforeseen combinations, or spend inordinate amounts of time working on the "wrong" problems. Too much of this and they give up, or post questions on online forums. Against this, many IF authors like to include in-story hints.

There are many approaches, which differ on two main issues.

First: do we spontaneously offer help to the player? The difficulty here is detecting the player's need: **Y ask Y?** tries to spot aimlessness, while **Solitude** has a novice mode where it is reasonable to assume that help is almost always needed. On the other hand, suppose we require that the initiative come from the player. Will a novice know to type HELP? **Query** shows how to redirect any attempt to ask a direct question into a HELP request. At the other end of the scale, wearily experienced players may type HELP all the time, out of habit, cheating themselves of the fun of frustration: if so, **Real Adventurers Need No Help** provides the nicotine patch against this addiction.

Second: how do we decide what help is needed? Normally the player only types HELP, which is unspecific. The simplest approach offers a menu, diagnosing the player's problem by obliging him to make choices: see **Food Network Interactive**. Listing all the possible problems in the story may give away too much, though, since players may not have reached the puzzles in question yet; so some authors prefer to create menus that adapt to the current state of the story (commonly called "adaptive hints").

Failing this, we can also try to parse commands like HELP ABOUT MICRODOT, as in **Ish. Trieste** takes a similar tack, except that instead of offering hints about puzzles, it offers help on story features (such as how to save), and lists all the available topics if the player types simply HELP.

Finally, and perhaps most stylishly, we can try to deduce what the player is stuck on from his immediate circumstances and from what is not yet solved: this needs a powerful adaptive hints system like the one in **The Unexamined Life**.

See **Getting Started with Conversation** *for a way to redirect a player using the wrong conversation commands*

See **Footnotes** *for another medium by which hints could perhaps be transmitted*

Start of Chapter 11: Out Of World Actions and Effects
Back to §11.2. Saving and Undoing
Onward to §11.4. Scoring

Example 111: ★ **Y ask Y?** Noticing when the player seems to be at a loss, and recommending the use of hints.

Example 284: ★ **Food Network Interactive** Using a menu system from an extension, but adding our own material to it for this game.

Example 296: ★ **Ish.** A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

Example 329: ★ **Query** Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

Example 285: ★★ **Trieste** Table amendment to adjust HELP commands provided for the player.

Example 400: ★★ **Solitude** Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the scenery.

Example 54: ★★★ **Real Adventurers Need No Help** Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

Example 230: ★★★ **The Unexamined Life** An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a nudge he wants at any given moment.

---

§11.4. Scoring

Not every work of IF allots a numerical score to the player: for some authors, this emphasises the idea of a story rather than a narrative. The simple sentence

```
Use scoring.
```

introduces the concept. Once this is included, Inform will provide built-in support for a single number measuring progress ("score"), and will expect to measure this against a maximum possible ("maximum score", which can either be set by hand or worked out automatically from a table of ranks).

In a story in which scoring exists, the player may choose to turn score notifications (such as "[Your score has just gone up by one point.]") on or off. The commands to do this are NOTIFY ON and NOTIFY OFF; the actions are called switching score notification on and switching score notification off. In the event that we need to amend the behavior of notification, we could do so by adding, removing, or modifying the elements of the check and carry out rulebooks for these commands; as in

```
Check switching score notification off:
    if the turn count is less than 10:
        say "You are still a novice, grasshopper. Allow your teacher to
give you advice until such time as you are ready to go on alone."
```

If we wish to change the wording of the default message ("[Your score has..."), we may want to use the Responses system.

An especially insidious style of bug allows the player to type the same sequence of

commands over and over, earning score endlessly for the same insight, and to avoid this it is usually safest to write source like:

```
After taking the Picasso miniature when the Picasso miniature is not
handled:
    increase the score by 10;
    say "As they say in Montmartre: dude!"
```

We might also write our condition with "for the first time", like so:

```
After jumping for the first time:
    increase the score by 5;
    say "Boing! That was certainly entertaining."
```

But we should be careful not to use "for the first time" in scoring situations where it's possible for the player to try the action but fail. Inform counts even unsuccessful attempts towards the number of times an action is understood to have occurred, so if the player tries to jump and fails, his "for the first time" will be used up and he will never receive the score points.

If there are many "treasure" items like the Picasso miniature, it is best to be systematic, as in **No Place Like Home**. **Bosch** takes another approach to the same idea, by creating a table of point-earning actions that the player will be rewarded for doing; the FULL SCORE command will then play these back.

Mutt's Adventure demonstrates how we might add a scored room feature, such that the player earns a point when he first arrives at a special room.

A single number does not really sum up a life, or even an afternoon, and **Goat-Cheese and Sage Chicken** and **Panache** offer more detailed citations. Works that are more story than story may prefer to offer a plot summary of the player's experience to date in lieu of more conventional scoring.

Finally, **Rubies** provides a scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next.

Start of Chapter 11: Out Of World Actions and Effects

Back to §11.3. Helping and Hinting

Onward to §11.5. Settings and Status Checks During Play

Example 219: **Bosch**

Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports.

Example 136: **Mutt's Adventure** Awarding points for visiting a room for the first time.

Example 137: **No Place Like Home** Recording a whole table of scores for specific treasures.

Example 166: **Panache** Replacing the score with a plot summary that records the events of the plot, scene by scene.

Example 280: **Goat-Cheese and Sage Chicken** Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this chapter.

Example 442: **Rubies** A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

---

§11.5. Settings and Status Checks During Play

Several default actions allow the player some control over the presentation of the story, or permit the player to request information about what is going on. In addition to the standard commands described elsewhere in this section (SCORE, SAVE, UNDO, QUIT, RESTART, and RESTORE), Inform has the following actions that control the player's experience:

Preferring abbreviated room descriptions (SUPERBRIEF)
Preferring unabbreviated room descriptions (VERBOSE)
Preferring sometimes abbreviated room descriptions (BRIEF)
Switching score notification on (NOTIFY ON)
Switching score notification off (NOTIFY OFF)

The first three of these allow the player to change the way rooms are described on first and subsequent versions; the last two, when used in a story that provides a score feature, toggle on and off reports such as "[Your score has just gone up by three points.]" These are discussed elsewhere in the Recipe Book (see cross-references below).

These provide immediate feedback about the status of the story file being played:

Verifying the story file (VERIFY)
Requesting the story file version (VERSION)
Requesting the pronoun meanings (PRONOUNS)

VERIFY examines checksums to make sure that the story file being run is intact and correct. This is less often an issue now than it was in the days when story files were distributed by highly corruptible floppy disk, but the command persists and is very occasionally useful. VERSION gives the full banner text associated with the story, including title, author, release number, IFID, and other bibliographical data; it follows this with a list of the included extensions.

PRONOUNS announces to the player what the story is currently understanding as the antecedents of "him", "her", "it", and "them". This is often useful during testing, but sometimes also during play.

The following allow the player (when supported by his interpreter) to create a log of play:

Switching the story transcript on (TRANSCRIPT ON)
Switching the story transcript off (TRANSCRIPT OFF)

It is rarely a good idea to change the default performance of such commands: they are often finicky and closely tied to the interpreter in which the story runs. Moreover, disabling the "version" command means that the story file is not able to display attribution information for Inform and any included extensions, in violation of their respective licenses.

See **Looking** *for a way to set the story's verbosity level for the player*

See **Scoring** *for a discussion of score notification*

See **Testing** *for some examples of status-check commands created for alpha- or beta-testing of a story*

Start of Chapter 11: Out Of World Actions and Effects
Back to §11.4. Scoring
Onward to §11.6. Ending The Story

§11.6. Ending The Story

Play can end in many ways, at the writer's discretion:

```
end the story;
end the story finally;
end the story saying "You have reached an impasse, a stalemate";
end the story finally saying "You have succeeded.";
```

The phrase "end the story" by itself will finish play, printing "*** The End ***". Using one of the phrases with "saying..." allows us to specify some other text with which to conclude. Including "finally" means that the player has earned access to AMUSING text and other notes, if any of these are provided.

We can eliminate the asterisked headline entirely by removing the rule that prints it, thus:

```
The print obituary headline rule is not listed in any rulebook.
```

The next step is to print the player's score and, if applicable, the rank he achieved. By default a story doesn't feature scoring, but the following use option will incorporate it:

```
Use scoring.
```

Then, if we want to allow a score but alter the way it is reported, we may remove or modify the print final score rule, as in

The print final score rule is not listed in any rulebook.

or perhaps something like

The chatty final score rule is listed instead of the print final score rule in for printing the player's obituary.

This is the chatty final score rule: say "Wow, you achieved a whole [score in words] point[s] out of a possible [maximum score in words]! I'm very proud of you. This was a triumph. I'm being so sincere right now."

What happens next is normally that the player is invited to RESTART, RESTORE (from a saved story), QUIT or UNDO the last command. The presence of the question can somewhat undercut a tragedy, and **Battle of Ridgefield** shows another way to go out.

If we do leave the question in, the text is formed by the Table of Final Question Options, which by default looks like this:

Table of Final Question Options

final question wording	only if victorious	topic	final response rule	final response activity
"RESTART"	false	"restart"	immediately restart the VM rule	--
"RESTORE a saved story"	false	"restore"	immediately restore saved story rule	--
"see some suggestions for AMUSING things to do"	true	"amusing"	--	amusing a victorious player
"QUIT"	false	"quit"	immediately quit rule	--
"UNDO the last command"	false	"undo"	immediately undo rule	--

Because this is a table, we may alter the behavior by changing entries or continuing the table. **Finality** shows how we might take out the option to UNDO the last command, for instance.

Using an ending phrase that includes "finally" tells Inform to include the options that are marked "only if victorious". One common use is to let the player read some special bit of additional text, perhaps describing easter eggs he might have missed in the story or presenting some authorial notes. **Xerxes** demonstrates a simple AMUSING command to read final information, while **Jamaica 1688** shows how to add completely new elements to the list of options.

Old-school adventures expected their adventurers to die early and die often. **Labyrinth of Ghosts** shows how the residue of such past attempts can be preserved into subsequent attempts, using an external file. **Big Sky Country** shows how a player can be resurrected by, let us say, some beneficent god, so that a player can even die more than once in the same attempt.

Start of Chapter 11: Out Of World Actions and Effects

Back to §11.5. Settings and Status Checks During Play

Onward to Chapter 12: Typography, Layout, and Multimedia Effects: §12.1. Typography

Example 383: ★ **Battle of Ridgefield** Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

Example 384: ★ **Finality** Not mentioning UNDO in the final set of options.

Example 385: ★ **Jamaica 1688** Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

Example 386: ★★ **Xerxes** Offering the player a menu of things to read after winning the game.

Example 441: ★★ **Labyrinth of Ghosts** Remembering the fates of all previous explorers of the labyrinth.

Example 138: ★★★ **Big Sky Country** Allowing the player to continue play after a fatal

accident, but penalizing him by scattering his possessions around the game map.

Examples from Chapter 11: Out Of World Actions and Effects

Start of this chapter
Chapter 12: Typography, Layout, and Multimedia Effects
Indexes of the examples

382 **Example Bikini Atoll** *WI*
Delaying the banner for later.

"Bikini Atoll" by Edward Teller

The Hut and the Tropical Beach are rooms.

The conch shell is in the Hut. After taking the shell for the first time: say "As you gather the oddly-warm conch shell into your arms, you experience a sudden flash of deja-vu...[banner text]"; move the player to the Tropical Beach.

Rule for printing the banner text when the player is not carrying the shell: do nothing.

Test me with "look / examine shell / get shell / look".

(By tradition, and as a courtesy to all the people who have worked on Inform, authors ensure that the banner is printed some time near the beginning of each game played. So please only defer it, rather than suppress it altogether.)

Example **Alien Invasion Part 23**

Keeping a preference file that could be loaded by any game in a series.

Suppose that we have a series of games each of which allows the player to select a puzzle difficulty level. When the player plays a new game in the series, we want him to start out by default with the same difficulty level he faced earlier on, so we store this information in a small preferences file, as follows:

```
"Alien Invasion Part 23"

A difficulty is a kind of value. The difficulties are easy,
moderate, hard, and fiendish.

Understand "use [difficulty] puzzles" as selecting
difficulty. Selecting difficulty is an action out of world,
applying to one difficulty.

Carry out selecting difficulty:
    choose row 1 in the Table of Preference Settings;
    now challenge level entry is difficulty understood;
    say "Puzzles will be [challenge level entry] from now
on."

The File of Preferences is called "prefs".
```

```
When play begins:
    if File of Preferences exists:
        read File of Preferences into the Table of
Preference Settings;
        choose row 1 in the Table of Preference Settings;
        say "(The current puzzle difficulty is set to
[challenge level entry].)"



Check quitting the game:
    write File of Preferences from the Table of Preference
Settings.



Table of Preference Settings
challenge level
easy



The Sewer Junction is a room.
```

Our preference file is restricted to a single option here for simplicity's sake, but we could keep track of more information -- whether the player preferred verbose or brief room descriptions, screen configurations, and so on.

If we were disposed to be somewhat crueler, we could use a similar method to make the player finish each episode of the series in order to "unlock" the next. All we would need to do is store a numerical password in our preferences file when the player finished a given level; the next level would check, say, the Table of Completed Levels for that password, and refuse to play unless the right number were present.

It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

Suppose we want a game in which each scenario starts with the characters wearing hats -- randomly passed out. We might be tempted to write our scenario like this:

```
"Hatless"

The Costumery is a room. Larry, Curly, and Moe are
men in the Costumery. Janine is a woman in the
Costumery.

Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
    say "[The target] is here, looking stylish in [an attire]."

Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."

A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.

The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.

When play begins:
    now every hat is worn by a random hatless person.
```

And we might hope that this would choose a new hatless person for each hat, but we would be wrong. It will instead choose one hatless person and put all the hats on him -- and everyone else has to go bare-headed. That's clearly no good. Let's try again:

```
"Hatless 2"
```

```
The Costumery is a room. Larry, Curly, and Moe are
men in the Costumery. Janine is a woman in the
Costumery.


Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
    say "[The target] is here, looking stylish in [an attire]."


Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."


A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.


The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.


When play begins:
    now every hatless person wears a random hat.
```

But this selects one random hat and assigns it to each hatless person in turn -- so it will only wind up being worn by the last of them (since Inform knows that only one person can wear a hat at a time).

In this case, we do have to expand out our loop so that the game makes an explicit distribution:

```
"Hatless 3"


The Costumery is a room. Larry, Curly, and Moe are
```

```
men in the Costumery. Janine is a woman in the
Costumery.


Rule for writing a paragraph about a person (called the
target) who wears a hat (called attire):
    say "[The target] is here, looking stylish in [an attire]."


Rule for writing a paragraph about a hatless person
(called the target):
    say "[The target] mopes about, hatless."


A hat is a kind of thing. A hat is always wearable.
Definition: a person is hatless if he is not the player and
he does not wear a hat.


The indigo bowler, the polka-dotted fedora, the pink
beret, and the scarlet cloche are hats.


When play begins:
    repeat with item running through hats:
        now the item is worn by a random hatless person.
```

Each time Inform considers the instruction "now the item is worn by a random hatless person", there is one fewer such person to choose from -- so we can guarantee that the hats are distributed one per customer and that all hats are distributed.

Hatless 3 is only guaranteed to work because the number of hats is less than or equal to the number of people; otherwise the final use of random will return "nothing" and then a problem message will appear during play.

P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

The answer is easy, but there is a trap:

```
Check saving the game when the location is the Vault:
say "That spell does not work here." instead.
```

The trap is that "Before saving the game...", which might have been our first guess, does not work: because out of world actions are exempt from Before, Instead and After rules.

"Spellbreaker" pulls this unpleasant, but in context witty, stunt as part of a situation which is engineered to force the player to reason through a weighing-objects puzzle using the perfect strategy rather than by guesswork. The illusion that the situation is fair - not rigged against the player, that is - would collapse if the player could save the game and keep retrying possibilities in the light of knowledge gained from earlier attempts. The moral of this story is that any attempt to use in-world situations to influence out-of-world commands should be *extremely uncommon*.

Example **A point for never saving the game** *WI*

In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

Here is one way to score this point with Inform:

```
Check saving the game for the first time: decrement the
score.
```

That has the right effect, but it just isn't sneaky enough. Instead let us quietly keep track of how many times the player saves:

```
Check saving the game: increment the number of
saves.
When play ends: if the number of saves is 0, increment
```

```
the score.
```

Sneakier, certainly, but now we could get the bonus even if the game ends earlier on, so finally:

```
When play ends: if the number of saves is 0 and the
score is 349, increment the score.
```

---

111 **Example Y ask Y?** *WI*

Noticing when the player seems to be at a loss, and recommending the use of hints.

Suppose we'd like to watch for signs that the player is floundering, and if we see them, recommend that he try the hints. There are probably more sophisticated diagnostics, but as a first cut, let's assume that a player who repeatedly reviews descriptions of objects he's already seen, looks around the room, and takes inventory, is at a loss for more productive activities. So then...

```
"Y ask Y?"

A thing can be examined or unexamined. A thing is
usually unexamined. Carry out examining something:
now the noun is examined.

Taking inventory is acting confused. Looking is acting
confused. Examining an examined thing is acting
confused.

After acting confused for the sixth turn:
    say "(If you are feeling lost, try typing HELP for
suggestions.)"
```

And now we write a scenario which will, alas, rather encourage even a

deft and clueful player to play as though he were hopelessly confused:

The story headline is "or: Bad Author, No Biscuit".

The description of a thing is usually "Hm. [The item described] reminds you quite a lot of [a random visible thing which is not the item described]."

The Yurt is a room.

Food is a kind of thing. Food is always edible. In the Yurt are a yam and a dish of yakitori. The yam and the yakitori are food. The description of food is "Well, at least it's not [a random edible thing which is not the item described]."

In the Yurt is an animal called a yapok.

The player wears a yukata. The player carries a yataghan.

Every turn:
    if a random chance of 1 in 2 succeeds and something is examined:
        say "Your eye is attracted by some kind of surreptitious movement from [the random examined thing].";
    otherwise if the player carries something and a random chance of 1 in 3 succeeds:
        say "[The random thing carried by the player] tries to slip from your grasp."

```
Test me with "x yam / x yam / look / x yam / i / look / i /
help / quit".
```

And finally a little dollop of perversity from a later chapter:

```
Check quitting the game:
    say "You're sure? ";
    if player consents, say "[line break]You were getting
close to a breakthrough, you know.[line break]";
    otherwise stop the action.


Understand "help" as a mistake ("You're doing fine! Just
keep at what you're doing now.").
```

Example **Food Network Interactive**

Using a menu system from an extension, but adding our own material to it for this game.

"Basic Help Menu" is an extension that uses tables to provide a menu of instructions. Suppose we wanted to include this menu in our own game, but add a few custom menu items of our own:

```
"Food Network Interactive"


Include Basic Screen Effects by Emily Short. Include
Menus by Emily Short. Include Basic Help Menu by
Emily Short.


Table of Basic Help Options (continued)

[illegible]
"Tales of Recipes"
"If you have any difficulties with [story title], please contact me at
fakeaddress@gmail.com."
```

This table is one that is pre-defined by the extension. By continuing it, we add a few additional items to the list.

And since we've promised a sub-menu of recipes:

```
Table of Recipes

"Fish, or tartare" buy extremely fresh salmon. Raw fish should be
served on the day it was caught, if possible. To guarantee this,
visit an Asian supermarket or specialty store, and buy salmon
marked 'sashimi grade'..."
"Begin by shelling half a pound of pecans..."

Whole Foods is a room.
```

To test it, type HELP and then experiment.

ExampleIsh.

A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

```
"Ish."

Ichiro's Dubious Sushi Hut is a room. "Despite the allure
of the dusty plastic sushi models in the window, you're
beginning to have second thoughts about the selection
of this particular restaurant for your rendezvous with
Agent Fowler. There are no other patrons, for one thing.
Afternoon sunlight filters lazily through the window and
illuminates a number of empty glass-topped tables, at
each of which is a chopstick dispenser (in form of cute
ceramic cat) and a pitcher of soy sauce (sticky).
```

```
The sushi bar itself is what gives the most pause,
however. Behind it sits an angry-looking Japanese
woman, aggressively eating a Quarter Pounder with
Cheese."
```

We can, when necessary, accept any text at all as a token:

```
Understand "help [text]" or "help about [text]" as getting
help about. Understand the commands "instructions" or
"hint" or "hints" or "menu" or "info" or "about" as "help".


Getting help about is an action applying to one topic.
```

After that, we can use "the topic understood" to refer to the text we read:

```
Carry out getting help about:
    if the topic understood is a topic listed in the Table of
Standard Help:
        say "[explanation entry][paragraph break]";
    otherwise:
        say "You're out of ideas."


Table of Standard Help

topic                explanation
"sushi"              "Sushi is just vinegar rice that sometimes has raw fish in the middle, and as long as you don't go too crazy with the wasabi, all should be well."
"California Roll"    "The California Roll is a concept"
"Fowler"             "Fowler's having this unusual delicacy is near zero."
"Wendy"              "Wendy will come here to deliver later. If Fowler ever turns up. Where is she, anyway?"
```

Since the player may not know what all the help options are, we might as well let him get an overview, as well.

```
Understand "help" as summoning help. Summoning
help is an action applying to nothing.


Carry out summoning help:
    say "Help is available about the following topics.
Typing HELP followed by the name of a topic will give
further information.[paragraph break]";
    repeat through the Table of Standard Help:
        say " [title entry]: [summary entry][line break]".


Test me with "help / help about microdot / help
cucumber / help california roll".
```

---

329 ExampleQuery *WI*

Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

First, we create a single "[query]" token so that we can capture all instances of such sentences in a single line:

```
"Query"


Blank Room is a room.


Understand "who" or "what" or "when" or "where" or
"why" or "how" or "who's" or "what's" or "when's" or
"where's" or "why's" or "how's" as "[query]".


Understand "[query] [text]" as a mistake ("[story title]
```

understands commands, such as '[command prompt]examine [a random thing that can be seen by the player]', but not questions. For more instructions, type HELP.").

Test me with "who am I? / who are you? / where is this place?".

Now the game will respond to all questions novice players might type with this reminder to look for help information.

Table amendment to adjust HELP commands provided for the player.

Suppose we are using an extension in which another author has defined some help topics for the player, and we want to amend them for our game.

We'll start with the portion of the text that we have inherited from the extension:

"Trieste"

Section 1 - Procedure

A help-topic is a kind of value. Some help-topics are defined by the Table of Standard Instructions.

Table of Standard Instructions

help-topic | "This game recognizes 150 common commands for forms of military attack. These include..."

```
"To save the game, type SAVE. You will be prompted to supply a
file-name for your saved game. If you'd like to return to play at that
point again later, RESTORE the saved game."
```

```
Understand "help [help-topic]" as asking for help about.
Asking for help about is an action out of world, applying
to one help-topic.
Understand "help" or "help [text]" as a mistake ("Help is
available on the following topics: [help-topics list]").
```

```
To say help-topics list:
    repeat through the Table of Standard Instructions:
        say "[line break] [help-topic entry]";
```

```
Carry out asking for help about:
    repeat through the Table of Standard Instructions:
        if the help-topic understood is the help-topic entry:
            say "[reply entry][paragraph break]";
            break.
```

```
Section 2 - Scenario
```

Now, let's imagine our game is a special one in which only a very limited supply of moves are allowed. In that case, we'll want to replace the information on commands:

```
Table of Standard Instructions (amended)

help-topic                  reply
"The only commands this game recognizes are HOLD, MOVE,
CONVOY, SUPPORT MOVE, and SUPPORT HOLD. No others
are necessary."
```

```
Board Room is a room. Mark is a man in the Board
```

Room. "Russia (played by Mark) is also hovering over the board."

Guest Bathroom is south of Board Room. Lena and Rob are in the Guest Bathroom. Lena is a woman. Rob is a man.

Rule for writing a paragraph about Lena when Lena is in the Guest Bathroom and Rob is in the Guest Bathroom:
    say "[Lena] (Italy) and [Rob] (Great Britain) are having a hushed conversation while leaning against your good towels. They stop and stare at you when you come in."

Test me with "help / help commands / help saving".

ExampleSolitude

WI

400. Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the scenery.

Observation of novice IF players suggests that they often have a hard time figuring out how to get started, especially if they are encountering the game in a context where they don't have time to settle in and read instructions. Here we provide some training wheels to help them learn to communicate.

This is divided into several parts. The first part is the system of rules for general guidance, which could be excerpted and used anywhere. The second part is a scenario using these rules.

"Solitude"

Part 1 - General Rules

```
When play begins:
    say "Have you played interactive fiction before? >";
    now novice mode is whether or not the player
consents.
```

The rationale for asking the question this way, and not another, is that novices asked whether they would like instructions very often say no, even if they need them.

```
Novice mode is a truth state that varies. Novice mode is
true.


Stopping novice mode is an action out of world.
Starting novice mode is an action out of world.


Understand "novice mode off" or "novice off" as
stopping novice mode.
Understand "novice mode on" or "novice on" as starting
novice mode.


Carry out stopping novice mode: now novice mode is
false.
Carry out starting novice mode: now novice mode is
true.


Report stopping novice mode: say "Novice mode is now
off."
Report starting novice mode: say "Novice mode is now
on."


Before reading a command when novice mode is true:
    say "[line break]Some options to try:[line break]";
    follow the novice suggestion rules.
```

The novice suggestion rules is a rulebook.

A novice suggestion rule (this is the suggestion that he look rule):
    if not looking and not going, say " [bold type]look[roman type]".

A novice suggestion rule (this is the suggestion that he check inventory rule):
    if the player carries something and we are not taking inventory, say " [bold type]inventory[roman type] (I)".

A novice suggestion rule (this is the suggestion that he put things on rule):
    if the player carries something and a free-standing supporter is relevant, say " [bold type]put[roman type] something [bold type]on[roman type] [the list of relevant supporters]".

A novice suggestion rule (this is the suggestion that he take things rule):
    if a gettable thing is relevant, say " [bold type]take[roman type] [the list of gettable relevant things]".

A novice suggestion rule (this is the suggestion that he examine things rule):
    if an unexamined thing is relevant, say " [bold type]examine[roman type] (X) [the list of unexamined relevant things]".

A novice suggestion rule (this is the suggestion that he enter things rule):
    if a relevant thing is worth entering, say " [bold

type]enter[roman type] [the list of worth entering relevant things], or [bold type]get out[roman type]".

A novice suggestion rule (this is the suggestion that he open things rule):
    if an unlocked openable thing is relevant, say " [bold type]open[roman type] or [bold type]close[roman type] [the list of unlocked openable relevant things]".

A novice suggestion rule (this is the suggestion that he lock things rule):
    if a closed lockable thing is relevant, say " [bold type]lock[roman type] or [bold type]unlock[roman type] [the list of closed lockable relevant things]".

A novice suggestion rule (this is the suggestion that he eat things rule):
    if the player carries an edible relevant thing, say " [bold type]eat[roman type] [the list of edible relevant things carried by the player]".

A novice suggestion rule (this is the suggestion that he wear things rule):
    if the player carries a wearable relevant thing, say " [bold type]wear[roman type] [the list of wearable relevant things carried by the player]".

A novice suggestion rule (this is the suggestion that he turn things on rule):
    if a device is relevant, say " [bold type]turn on[roman type] or [bold type]turn off[roman type] [the list of relevant devices]".

A novice suggestion rule (this is the suggestion that he go places rule):
    if a room is adjacent, say " [bold type]go[roman type]

```
[exit list][if in darkness] or try other directions in the
dark[otherwise]".
```

```
A novice suggestion rule (this is the suggestion that he
enter doors rule):
    if an open door is relevant, say " [bold type]go
through[roman type] [the list of relevant open doors]".
```

```
A novice suggestion rule (this is the suggestion that he
interact with people rule):
    if another person is relevant, say " [bold
type]kiss[roman type] or [bold type]wake[roman type]
[the list of relevant other people][if the player carries
something], or [bold type]give[roman type] things [bold
type]to[roman type] someone[end if]".
```

```
A novice suggestion rule (this is the suggestion that he
ask for help rule):
    say " [bold type]help[roman type] to see a more
complete set of instructions".
```

```
A novice suggestion rule (this is the suggestion that he
turn off help rule):
    say " [bold type]novice mode off[roman type] to turn
off this guidance".
```

```
Last novice suggestion rule:
    say "[line break]".
```

The suggestion about asking for help is no good unless we provide some. This might take any of a number of forms, but for the sake of example we'll use an easy way out:

```
Include Basic Screen Effects by Emily Short. Include
Menus by Emily Short. Include Basic Help Menu by
Emily Short.
```

After taking inventory when novice mode is true: say "To get rid of any of these objects, [bold type]drop[roman type] it."

A thing can be examined or unexamined. Carry out examining something: now the noun is examined.

A thing can be seen or unseen. A thing is usually unseen.

Definition: a thing is relevant if it is seen and it is visible.
Before printing the name of something (called the target): now the target is seen; if novice mode is true, say "[bold type]". After printing the name of something: say "[roman type]".

Definition: a supporter is worth entering:
    if the player carries it, no;
    if it is enterable, yes.

Definition: a container is worth entering:
    if the player carries it, no;
    if it is enterable and it is open, yes.

Definition: a person is other if it is not the player.
Definition: a person is another if it is other.

Definition: a thing is free-standing if it is in a room.

```
To say exit list:
    let place be location;
    let count be 0;
    repeat with way running through directions:
        let place be the room way from the location;
        if place is a room:
            increment count;
            say "[if count is greater than 1] or[end if] [bold
type][way][roman type]".
```

```
Definition: a thing is gettable:
    if it is scenery, no;
    if it is fixed in place, no;
    if it is a person, no;
    if the player is carrying it, no;
    if the player is wearing it, no;
    yes.
```

Part 2 - On the Ground

Antarctic Research Station is a room. "Though not always the most stimulating of environments, the station is far from your ex-wife and most of the things in the world that annoy you, namely the other 6+ billion people. There is a second room to the south." The station contains a radio. The radio is a device. It is fixed in place.

South of the Station is Sitting Room. The description of the Sitting Room is "Just big enough for a very [comfortable chair]." The Sitting Room contains an enterable supporter called a comfortable chair. The chair is scenery. A monograph about penguins is in the Sitting Room.

Blistering Cold is a room. "It is white out here and very very very cold." The white door is a door. "[The white door] leads to [the other side of the white door]." It is west of the Blistering Cold and east of the Antarctic

Research Station.

Test me with "i / x radio / x door / s / i / x chair / x monograph / sit in chair / get up / n / open door / enter door".

ExampleReal Adventurers Need No Help *WI*

Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

Suppose we have an action called "asking for help" that gives the player some hints on request. We've also made it possible to turn this feature off, if the player would like to discourage himself from using the hints too much. Now we need a value that varies to keep track of whether hints are currently permitted or currently not permitted. So we might write:

```
"Real Adventurers Need No Help"

A permission is a kind of value. The permissions are
allowed and denied.

Hint usage is a permission that varies. Hint usage is
allowed.
```

And under the right circumstances, we change hint usage to denied:

```
Check asking for help for the first time:
    say "Sometimes the temptation to rely on hints
becomes overwhelming, and you may prefer to turn off
hints now. If you do so, your further requests for
guidance will be unavailing. Turn off hints? >";
    if player consents:
        now hint usage is denied;
        say "[line break]Truly, a real adventurer does not
```

```
need hints." instead.
```

Then we can refer back to this value later to decide whether we want to display the hint menu or not:

```
Check asking for help:
    if hint usage is denied, say "You have chosen to
eschew hints in this game. Be strong! Persevere!"
instead.


Asking for help is an action out of world. Understand
"help" or "hint" or "hints" as asking for help.


The Realm of Terribly Unjust Puzzles is a room.


Carry out asking for help:
    say "Fine, since you're weak enough to ask: here is a
complete walkthrough: GET EGG. PEEL EGG. SMELL
EGG. DIVIDE YOLK INTO THREE PORTIONS. GIVE
THE SMALLEST PORTION OF YOLK TO THE
GOLDEN GOOSE. ASK THE GOOSE ABOUT
WHETHER THE SWAN IS TO BE TRUSTED. GIVE
THE LARGEST PORTION OF YOLK TO THE SWAN.
DANCE CONGA. EAT MEDIUM PORTION. STAND ON
HEAD. WEST."


Test me with "hint".
```

Note that it would probably be kinder to offer the player some intermediate level of help, in the actual event.

An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a

nudge he wants at any given moment.

Hint systems in IF come in a variety of flavors: some are a static, prewritten set of guidelines (which might exist in a menu or outside the game entirely); others are built in as part of the program, and attempt to adapt to the situation the player currently faces. Adaptive hints have the advantage that they are less likely to reveal information for which the player is not ready, and the disadvantage that they are more work for the author.

The exercise here is to write an adaptive hint system that will both respond in agile ways to the state of the world model and require a minimum of authorial fussing. We also want the player to be able to ask for a hint about any object he encounters in the game world: this will let him be specific and avoid accidentally receiving hints about the wrong puzzles.

Our baseline assumption is that a player may find a puzzle unsolvable for one of two reasons: he either hasn't seen the relevant clue, or he hasn't got the relevant equipment. If these are true, then he should be given hints about how to find this information, and then once he has it, more specific hints about the puzzle itself -- ending, as a last resort, with the exact command(s) he will need to use in order to bring about the solution.

In practice, there are other possibilities, but this will do for an example.

We begin by defining our relations:

```
"The Unexamined Life"


Use scoring.


Explaining relates one thing to various things. The verb
to explain means the explaining relation.


Instead of hinting about something when something
unexamined (called the clue) explains the noun:
    say "You're still missing some information that might
be useful to understanding the problem. [More]";
```

```
    if player consents, try hinting about the clue.
```

```
Requiring relates one thing to various things. The verb
to require means the requiring relation.
```

```
Instead of hinting about something when the noun
requires something (called the implement) which is not
carried by the player:
    say "You're missing an object that might be useful to
resolving this problem. [More]";
    if player consents, try hinting about the implement.
```

```
Hinting about is an action applying to one visible thing.
Understand "hint about [any thing]" as hinting about.
```

This allows us to create the most absolutely generic sort of hint -- boring, perhaps, but in practice the player often just needs a nudge about what part of the game world he should be examining for a solution:

```
Carry out hinting about:
    if something explains the noun, say "You might want
to review [the list of things which explain the noun]. ";
    if the noun requires something:
        say "You should be sure that you have [the list of
things required by the noun]. ";
    otherwise:
        say "Sorry, I can't advise you further on that.".
```

These things cover hinting about objects that are themselves puzzles. But what if the player asks for hints about a tool or piece of information because he doesn't know how to apply it yet? We might want to give some guidance there, as well.

```
Carry out hinting about something which explains
something (called target):
    if target is unseen, say "[The noun] might prove
useful information, sooner or later." instead;
    otherwise say "You could examine [the noun]."
instead.
```

```
Carry out hinting about something which is required by
something:
    say "[The noun] might be useful to have. [More]";
    if player consents:
        if a seen thing requires the noun, say "[The noun]
may help with [the list of seen things which require the
noun]." instead;
        otherwise say "There are [number of things which
require the noun in words] problems for which [the noun]
might come in handy." instead.
```

Now we have these general hints written, but we want to pre-empt them if the player has not yet fulfilled all the prerequisites.

```
Instead of hinting about something unseen:
    if the noun is visible:
        now the noun is seen;
        continue the action;
    say "Perhaps you should explore further. ";
    if the ultimate location of the noun is an unvisited
room:
        try hinting about the ultimate location of the noun;
    otherwise:
        if the ultimate location of the noun is the location:
            say "You're in the correct room right now[if the
visible shell of the noun is a thing]. Try further exploring
[the visible shell of the noun][end if].";
        otherwise:
            try hinting about the ultimate location of the
noun.
```

```
Instead of hinting about a visited room:
    say "There's a room you've visited, but you haven't
exhausted all there is to see there. [More]";
    if player consents:
        say "Try going back to [the noun]. [More]";
        if player consents, direct player to the noun.
```

```
Instead of hinting about an unvisited room:
    say "There's a room you haven't yet visited. [More]";
    if player consents, direct player to the noun.
```

```
To direct player to (goal - a room):
    let way be the best route from location to the goal,
using even locked doors;
    if way is a direction, say "Try going [way] to start your
explorations.";
    otherwise say "Sorry, the route is an indirect one.".
```

```
Instead of hinting about a portable seen thing which is
not visible:
    if the noun is scenery, continue the action;
    say "You have seen the item you need to solve this
problem, but it's not in sight at the moment. [More]";
    if player consents:
        try hinting about the ultimate location of the noun.
```

And this business of "seen" things requires, of course, that we keep track:

```
A thing can be seen or unseen. A thing is usually
unseen. The player is seen. After printing the name of
something (called target): now the target is seen.
```

That "After printing..." rule means that as soon as the game automatically prints the name of an object, it tags that object as having been "seen" by the player. This requires just a little care on our part, that we never mention an object without using the game's printing rules. Still, it is much easier than most other possible forms of bookkeeping.

We also need to deal with the question of whether the player has examined an object, for those objects whose descriptions carry vital information:

```
A thing can be examined or unexamined. A thing is
usually unexamined. Carry out examining something:
now the noun is examined.
```

In practice, there might be other ways of getting vital facts, and in a more sophisticated puzzle game we might need a more sophisticated model to track this. But examined or unexamined will do for now.

```
To decide what room is the ultimate location of (item - a
thing):
    let place be the holder of the item;
    while the place is a thing:
        let the place be the holder of the place;
    if the place is a room, decide on the place.


To decide what thing is the visible shell of (item - a
thing):
    if item is visible, decide on the item;
    let place be the holder of the item;
    while place is a thing and place is not visible:
        let place be the holder of the place;
    if the place is visible, decide on the place.


To say more:
    say "[paragraph break]Shall I go on? > ".
```

That covers most of the generic hints, but let's also add some slightly more precise hints about a few kinds of objects that are especially important in the model world. These hints will probably not be very interesting to a seasoned IF veteran, but a novice player who does not know the wording or cannot guess what something might be for may still find them useful:

```
Carry out hinting about a locked lockable thing:
    say "You could unlock [the noun] with [the matching
key of the noun]." instead.


Instead of hinting about a locked thing when the
matching key of the noun is not carried by the player:
    if the player can see the matching key of the noun:
        say "Perhaps [the matching key of the noun] would
help.";
    otherwise:
        say "[The noun] is locked. There must be a key
around somewhere. [More]";
        if player consents, try hinting about the matching
key of the noun.
```

```
Carry out hinting about a closed openable unlocked
thing:
    say "You could open [the noun]." instead.

Carry out hinting about an open door:
    say "You could enter [the noun]." instead.

Carry out hinting about an unexamined thing:
    say "You might find out something if you examine [the
noun]." instead.

Carry out hinting about an edible thing:
    say "You could eat [the noun]." instead.

Carry out hinting about a wearable thing:
    say "You could wear [the noun]." instead.

Carry out hinting about a pushable between rooms
thing:
    say "You could push [the noun] some direction."
instead.
```

Now to the actual objects in the game:

```
The Crypt is a room. "This squat, barrel-vaulted
chamber runs roughly north-south. Along either side are
the graves of Saxon kings and early bishops of the
church long since gone to dust -- one [tomb] in particular
looks undisturbed."
```

Notice that we used the bracketed tomb here: the tomb is scenery, and if we do not use the name-printing function, Inform will not register that we have mentioned it to the player.

```
The tomb is scenery in the Crypt. The tomb is openable
and closed. The silver dagger is a thing in the tomb.
Understand "tombs" as the tomb. The description of the
silver dagger is "Gleaming in a soft light all its own. Its
blade is figured with running deer and its hilt is made of
horn." The wight requires the silver dagger. The tomb
requires the pry bar.


Instead of opening the tomb when the player does not
carry the pry bar:
    say "The lids are stone, too heavy for you to raise
without some implement."
```

Now we can add specific hints to replace the generic ones:

```
Carry out hinting about the tomb:
    say "The lids are heavy, but you can open them when
you carry the pry bar."
```

The rest of the hint system ensures that the player will not see this final suggestion until he has the pry bar, since the tomb "requires" the pry bar. Having the hint there doesn't excuse us from providing some alternate wording in case the player solves this not-very-difficult conundrum on his own, though:

```
Understand "pry [something] with [something preferably
held]" as unlocking it with. Understand the commands
"lever" or "prise" as "pry".


Instead of unlocking something with the pry bar, try
opening the noun.


The wight is a man in the Crypt. "[The wight] lurks near
the south exit." The description of wight is "Old English
[italic type]wiht[roman type]: a thing, a creature. It is little
more than the memory of a life ill-lived, but it lingers
here." Understand "wiht" or "creature" or "ghost" as the
wight.
```

Instead of going south in the presence of wight:
    say "The wight breathes chill into your face.

Your head swims, and you are aware that you no longer have the willpower to go in that direction."

Fresh Air is south from the Crypt.

After going to Fresh Air:
    increment the score;
    say "Congratulations, you have escaped!";
    end the story finally.

The inscription is fixed in place in the Crypt. "Someone has painstakingly carved [an inscription] into the wall above the door." The description is "Squinting, you decipher the Latin text: [italic type]Silver causes harm to those that live though dead[roman type]." The inscription explains wight.

The Treasure Chamber is north of the Crypt. "The walls are thick, the high windows promisingly barred with iron. But for all this there is no hint of any valuable stores remaining."

The pry bar is in the Treasure Chamber. "One of the window bars, rusted from its place, lies in a puddle of water." Understand "window" or "bars" as the pry bar. The description of the pry bar is "A few feet long, and not entirely rusted into uselessness yet."

Instead of giving the dagger to wight:

say "The wight recoils, appalled."

Carry out hinting about wight:
say "You will have to find some way to get wight to come in physical contact with the silver dagger, which he will certainly not do willingly. [More]";
if player consents, say "You could, for instance, throw it at him." instead;
otherwise stop the action.

Understand "touch [something] with [something]" as putting it on (with nouns reversed). Understand "hit [someone] with [something]" as putting it on (with nouns reversed).

Instead of attacking the wight:
say "You can't force yourself to approach close enough for hand to hand combat: if, indeed, the wight has hands."

Instead of putting the dagger on wight:
say "The wight fades out of your way without ever coming into contact with the dagger. Perhaps a more projectile method would work better."

Instead of putting something on wight:
say "The wight dodges you."

Instead of throwing the dagger at wight:
now the wight is nowhere;
move the dagger to the location;
increment the score;
say "The dagger passes through its airy form with a rending like the rip of silk. The fragments dissipate at once."

```
The maximum score is 2.


Test me with "hint about wight / north / get bar / south /
open tomb / get dagger / south / hint about wight / read
inscription / hint about wight / attack wight / throw
dagger at wight / south".
```

Note that, if using TEST ME to run through the solution on the Z-machine, we will have to answer a few yes/no questions along the way.

For Glulx, the code should instead read something like

```
Test me with "hint about wight / y / north / get bar /
south / open tomb / get dagger / south / hint about wight
/ y / read inscription / hint about wight / y / attack wight /
throw dagger at wight / south".
```

---

219 ExampleBosch *WI*

Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports.

We could, if we wanted, make a table of stored actions all of which represent things that will earn points for the player. For instance:

```
"Bosch"


Use scoring.


The Garden of Excess is a room. The gilded lily is an
edible thing in the Garden of Excess.
```

The Pathway to Desire is west of the Garden of Excess.
The emerald leaf is in the Pathway.

Table of Valuable Actions

[illegible]	[illegible]tion
15	taking the emerald leaf
5	eating the gilded lily

(And our list would presumably continue from there, in the full game.)

The maximum score is 25.

After doing something:
    repeat through Table of Valuable Actions:
        if the current action is the relevant action entry and turn stamp entry is less than 0:
            now the turn stamp entry is the turn count;
            increase the score by the point value entry;
    continue the action.

Understand "full score" or "full" as requesting the complete score. Requesting the complete score is an action out of world.

Check requesting the complete score:
    if the score is 0, say "You have not yet achieved anything of note." instead.

Carry out requesting the complete score:
    say "So far you have received points for the following:[line break]";
    sort the Table of Valuable Actions in turn stamp

```
order;
    repeat through the Table of Valuable Actions:
        if the turn stamp entry is greater than 0:
            say "[line break] [relevant action entry]: [point
value entry] points";
    say line break.



Test me with "eat lily / w / full score / get leaf / full".
```

This system is tidy, but limited: we cannot give actions interesting names in the score list, like "seducing the pirate's daughter" or "collecting a valuable artifact". So it will not be ideal in all situations, but it has the virtue of being easy to extend, and of listing all of the player's successes in the order in which they occurred in his play-through.

Awarding points for visiting a room for the first time.

Suppose we want to reward the player the first time he reaches a given room. The "unvisited" attribute is useful for this: unlike such constructions as "going to a room for the first time", it doesn't develop false positives when the player has merely tried to go to the room in question. "Every turn when the player is in a room for the first time" is also unhelpful, because it continues to be true as long as the player is in a room on his first visit there.

```
"Mutt's Adventure"



Use scoring.



Section 1 - Procedure



A room can be scored or unscored.
```

Carry out going to a unvisited scored room:
    increment the score.

Section 2 - Scenario

The Incan Palace Compound is a room. "After numerous false leads through the jungles of Peru, and an arduous trek along the Amazon, you have arrived, at last, here: at Atagon, the lost city of untold treasure."

The startlingly intricate door is a door. It is inside from Incan Palace Compound and outside from the Treasure Room. "A door carved all over with figures of ancient gods, and protected by an assortment of gears and latches, [if open]stands open[otherwise]blocks progress[end if] towards [the other side of the intricate door]."

The description of the Treasure Room is "To your considerable surprise, the treasure room is stocked with art objects from a vast range of eras and geographical locations: beside the expected pre-Columbian gold there are Cycladic figurines, Chinese Tang-dynasty pottery, purses that might have been stolen from Sutton Hoo. [one of]If the British Museum developed a nasty expectorant cough, this is what you'd find in its hanky. [or][stopping]".

The Treasure Room is scored.

Test me with "in / out / in".

Recording a whole table of scores for specific treasures.

Suppose we want to assign scores for a whole range of objects the player might pick up. One systematic way to do this would be with a table of point values for things:

"No Place Like Home"

Use scoring.

The Hall of the Gnome King is a room. The emerald cow is a thing in the Hall of the Gnome King. The ivory chessman is a thing in the Hall of the Gnome King. The book of incantations is a thing in the Hall of the Gnome King.

Table of Point Values

item	score
cow	10
incantations	4
chessman	1

Report taking an item listed in the Table of Point Values:
    increase the score by the score entry;
    blank out the whole row.

Test me with "take all".

"Blank out the whole row" removes the line from the table, so that each award will occur only once. The player will not be able to earn more and

more points by dropping and taking the same item again.

ExamplePanache *WI*

Replacing the score with a plot summary that records the events of the plot, scene by scene.

If we have a plot that branches and has multiple kinds of outcome, we might well want to assemble these into a plot summary in place of the more traditional score. One way to approach this is to build the scene information into a table, adding information when each scene ends.

We begin with a bit of setup:

"Panache"

The player is in a room called Beneath Roxane's Balcony. Christian is a man in the Balcony. "Christian stands in a spot of moonlight and tries to avoid too obviously glancing at the shadows that conceal you." The description of Christian is "Like you, Christian loves Roxane. Unlike you, he is handsome enough to receive her favor in return. He is the beauty to your brain."

Roxane is a woman in the Balcony. "Above you in the night is Roxane." Roxane can be wooed, skeptical, confused, or annoyed. Roxane is skeptical. The description of Roxane is "The brightest, the most radiant of women -- and in love with an utter fool."

Empty Street is a room. "No one is about at this hour, all alone under a pale moon."

Telling someone about something is speech. Asking someone about something is speech. Answering

someone that something is speech.

This next portion borrows from the Advanced Actions chapter to allow us to command Christian to do things:

A persuasion rule for asking Christian to try speech:
persuasion succeeds.

Carry out Christian answering someone that something:
    now Roxane is wooed;
    say "'[noun], [the topic understood].'"

Carry out Christian answering the player that something:
    say "Christian parrots your words back to you."
instead.

Carry out Christian telling a skeptical Roxane about something:
    now Roxane is confused;
    say "Christian turns to [the noun]. 'I must tell you about [the topic understood],' he says, and comes to a halt, looking at you for further direction.

    Perhaps you'd better give him exact lines to say. Surely he can't mess up an instruction like 'say hello to Roxane.'" instead.

Carry out Christian asking a skeptical Roxane about something:
    now Roxane is confused;
    say "'So,' says Christian nervously to [the noun]. 'Did you know about [the topic understood]?' But Roxane merely seems puzzled." instead.

```
Carry out Christian telling a confused Roxane about
something:
    now Roxane is annoyed;
    say "Christian begins rambling on witlessly about [the
topic understood]." instead.


Carry out Christian asking a confused Roxane about
something:
    now Roxane is annoyed;
    say "Christian puts another confused question about
[the topic understood]." instead.
```

And now we have enough material to begin writing the scenes:

```
Courting Roxane is a scene. Courting Roxane begins
when play begins. Courting Roxane ends in success
when Roxane is wooed. Courting Roxane ends in failure
when Roxane is annoyed.


When Courting Roxane ends in success:
    record "Seduction by Proxy" in the Table of Events;
    say "Roxane, deeply moved by this sentiment, invites
Christian up to her balcony. He scrambles up the ivy
and disappears into her bedroom; the last thing you
hear is a girlish giggle from above.";
    now Roxane is nowhere; now Christian is nowhere;
    move the player to Empty Street.


When Courting Roxane ends in failure:
    record "Ruining Christian's Chances" in the Table of
Events;
    say "Roxane sighs heavily and goes back into her
room, slamming the door behind her.


'Thanks very much,' says Christian to you, striding off
down the street.";
    now Roxane is nowhere; now Christian is nowhere;
    move the player to Empty Street.
```

Sulky Ramble is a scene. Sulky Ramble begins when Courting Roxane ends in success. Sulky Ramble ends when the time since Sulky Ramble began is 2 minutes. When Sulky Ramble ends: record "Wandering the Streets, Sulking" in the Table of Events.

Every turn during Sulky Ramble:
    say "You find yourself kicking fenceposts quite without thinking about it."

Smug Ramble is a scene. Smug Ramble begins when Courting Roxane ends in failure. Smug Ramble ends when the time since Smug Ramble began is 2 minutes. When Smug Ramble ends: record "Wandering the Streets, Exultant" in the Table of Events; say "Of course, you will regret this soon enough."

Every turn during Smug Ramble:
    say "You find yourself smiling fiercely at the moon."

To record (occurrence - text) in (target table - a table name):
    choose a blank row in the target table;
    now the event entry is the occurrence.

Table of Events
event
"A Duel of Insults"
with 30 blank rows.

The plot summary rule is listed instead of the announce the score rule in the carry out requesting the score rules.

```
This is the plot summary rule:
    say "The Plot So Far: [paragraph break]";
    let act number be 0;
    repeat through the table of Events:
        increment act number;
        say " Act [act number]: [event entry][line break]".


Test me with "christian, ask roxane about love /
christian, say your breath smells like ripe taleggio to
roxane / score / z / z / score".
```

Example Goat-Cheese and Sage Chicken *WI*

Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this chapter.

Some games provide a FULL SCORE command that gives more information about the player's achievements than SCORE alone. Supposing we wanted to include a FULL SCORE in our game that gave the kind of score reading described in this chapter:

```
"Goat-Cheese and Sage Chicken"


Use scoring.


The story headline is "An interactive recipe"


Table of Tasks Achieved

Task                    Points
Finding the recipe      3
Tasting onions"
```

3 "reconstituting apricots"
1 "flattening chicken"
1 "unwrapping goat cheese"

To record (T - text) as achieved:
    choose row with a citation of T in the Table of Tasks Achieved;
    if there is no time entry:
        now time entry is the time of day;
        increase the score by the points entry.

Requesting the full score is an action out of world. Understand "full" or "full score" as requesting the full score.

Carry out requesting the full score:
    if the score is 0, say "You have achieved nothing towards supper." instead;
    repeat through the Table of Tasks Achieved in reverse time order:
        say "[time entry]: [citation entry] ([points entry])."

Table of Rankings

Score	Rank
0	"Rank Amateur"
2	"would-be Bobby Flay"
5	"Alton Brown"
8	"Julia Child"

The Kitchen is a room. The description of the Kitchen is "Equipped with many familiar friends: refrigerator, stove, oven; countertop; cabinet for pans and bowls, and a drawer for your tools."

The stove is scenery in the kitchen. It is a supporter. The oven is a container. It is part of the stove. It is closed and openable. The stove's switch is a device. It is switched on. It is part of the stove. The oven's dial is a device. It is switched off. It is part of the oven.

A thing can be heatproof.

Instead of putting something which is not heatproof on the stove when the stove's switch is switched on:
    say "You catch yourself just at the last minute: not a good idea to put [the noun] directly on the stove while it's turned on."

Instead of switching on the stove, try switching on the stove's switch. Instead of switching off the stove, try switching off the stove's switch. Instead of switching on the oven, try switching on the oven's dial. Instead of switching off the oven, try switching off the oven's dial.

Before switching on the oven's dial when the oven is open:
    say "(closing the oven so that it will heat properly) [command clarification break]";
    try closing the oven.

The frying pan is a heatproof unopenable open container on the stove.

The cabinet is a closed openable container in the kitchen. It is scenery. It contains an open unopenable container called a mixing bowl. It contains a portable supporter called a platter. An open unopenable heatproof container called a Calphalon baking dish is in the cabinet. The baking dish has the description "One of those marvelous pieces of kitchen equipment which

goes on the stove or in the oven, as you will. The chief thing is never ever to touch it when it is hot, since the handles are metal and the heat retention excellent."

The counter is a supporter in the kitchen. It is scenery. The kettle is a heatproof openable closed container on the counter. Some water is in the kettle.

The water can be cool, warm, or boiling. The printed name of the water is "[water condition] water".

The refrigerator is a closed openable container in the kitchen. It is scenery. Understand "fridge" as the refrigerator.

An ingredient is a kind of thing.

Some onions, some apricots, and some sage are ingredients on the counter. A chicken breast, an egg, and goat cheese are ingredients in the refrigerator.

The goat cheese can be wrapped, snipped open, or unwrapped. The printed name of the goat cheese is "[goat cheese condition] goat cheese".

The sage can be unwashed, clean, or julienned. The sage is unwashed. The printed name of the sage is "[sage condition] sage".

The apricots can be dried, reconstituted, or chopped. The apricots are dried. The printed name of the apricots is "[apricots condition] apricots".

The chicken breast can be whole, flattened, stuffed, rolled, coated, browned, or baked. The printed name of the chicken breast is "[chicken breast condition] chicken breast".

The onions can be unpeeled, peeled, diced, sauteed, or burnt. [The printed name of the onions is "[onions condition] onions".]

The can of chicken broth is a closed container on the counter. The bottle of white cooking wine and the bottle of Thurston Wolfe PGV are a closed containers in the refrigerator.

The description of the Thurston Wolfe is "A Washington State Pinot Gris-Viognier, 2003. It is said to have 'peach aromas', and, startlingly, the untutored person can detect these without resorting to fantasy.

(It is also supposed to possess a delicate perfume and a moderate body; the label author at least stopped short of 'good sense of humor and likes long walks on the beach')."

Understand the commands "wash" and "rinse" as "clean".

Instead of rubbing the unwashed sage:
    now the sage is clean;
    say "You rinse off the sage. There -- ready to slice."

Instead of cutting the sage:

say "You'd need to have a knife in hand, first."

Instead of cutting the clean sage when the player is carrying the butcher knife:
now the sage is julienned;
say "You slice the sage into thin strips."

Instead of cutting the unwashed sage:
say "It came from the garden, so it won't have any strange chemicals on it, but you should still give it a rinse for dirt and bugs and so on before using it."

Instead of doing something other than examining or rubbing with the unwashed sage:
say "It needs to be washed off."

Understand "peel [something]" as peeling.

Peeling is an action applying to one thing.

Instead of peeling the unpeeled onions:
now the onions are peeled;
say "You tear away the shining outer skin of the onions, leaving them pale and nekkid. Poor things."

Instead of cutting the diced onions:
say "That seems unnecessary now."

Instead of cutting the sauteed onions:
say "Too late; you're well past that stage."

Instead of cutting the burnt onions:
    say "There's no rescuing 'em -- the carbon isn't going to flake off, you know."

Instead of cutting the unpeeled onions:
    say "It would help to peel them first."

Instead of cutting the peeled onions:
    say "You'd need to have a knife in hand, first."

Instead of cutting the peeled onions when the player is carrying the butcher knife:
    now the onions are diced;
    say "You dice the onions neatly. Your own skill brings tears to your eyes."

Instead of opening the goat cheese:
    try peeling the goat cheese instead.

Instead of peeling the unwrapped goat cheese:
    say "The goat cheese is already unwrapped. (Stay focused, stay focused...)"

Before peeling the wrapped goat cheese when the shears are held by the player:
    try cutting the goat cheese.

Instead of peeling the snipped open goat cheese:
    now the goat cheese is unwrapped;
    record "unwrapping goat cheese" as achieved;
    say "Ah, success. The goat cheese is now free of its packet."

Instead of peeling the wrapped goat cheese:
    say "It would help to have a pair of scissors or something -- the packet resists being torn."

Instead of cutting the goat cheese:
    say "No need, at this point."

Before cutting the wrapped goat cheese when the shears are not held by the player and the shears are visible:
    say "(first picking up the shears)[command clarification break]";
    try taking the shears.

Instead of cutting the wrapped goat cheese:
    say "Something to cut with would be useful."

Instead of cutting the wrapped goat cheese when the shears are held by the player:
    now the goat cheese is snipped open;
    say "You neatly snip through the packaging with the shears."

Instead of examining the whole chicken breast:
    say "It is still entire and has yet to be pounded flat."

Instead of examining the flattened chicken breast:
    say "It has been hammered to a thickness of about a half inch. (The recipe said a quarter inch but you're pretty sure it was joking. You have never been able to achieve a quarter inch.)"

Instead of attacking the whole chicken breast:
    say "You need something heavy enough to flatten it with."

Instead of attacking the whole chicken breast when the player is holding the wooden mallet:
    now the chicken breast is flattened;
    record "flattening chicken" as achieved;
    say "You hammer away at the chicken breast, turning all your aggressions into culinary goodness. Several minutes pass. When you are done you have a broad flat chickeny pancake suitable for wrapping about a stuffing."

Before printing the name of onions:
    say "[onions condition] ".

The drawer is an openable closed container. It is part of the counter.

A tool is a kind of thing. A spatula, a spoon, a wooden mallet, some shears, and a ball of twine are tools in the drawer. A butcher knife is a tool carried by the player. Understand "scissors" as the shears.

Instead of burning something:
    say "You'll have to do that the hard way."

Some steam is fixed in place. "Dense clouds of steam fill the room."

Some smoke is fixed in place. "Smoke is beginning to collect near the ceiling."

Sauteeing Onions is a scene. Sauteeing Onions begins when the diced onions are in a hot container.

Definition: a container is hot if it is on the stove and the stove's switch is switched on.

Instead of touching the hot pan:
    say "Ow!"

Scorching Onions is a scene.

Preheating the Oven is a scene. Preheating the Oven begins when the oven's dial is heating.

Definition: a oven's dial is heating if the oven's dial has been switched on for exactly one turn.

Preheating the Oven ends when the time since Preheating the Oven began is five minutes.

When Preheating the Oven begins:
    say "The oven begins to warm up."

When Preheating the Oven ends:
    say "The oven beeps to inform you that it has reached the desired hotness."

Every turn during Sauteeing Onions:
    say "The onions sizzle in the pan."

Every turn during Scorching Onions:
    say "The onions are past their prime and are getting blacker by the moment."

Every turn during Hearing the Kettle Whistle:
    say "The kettle continues to whistle."

Instead of listening to during Hearing the Kettle Whistle:
    say "The only thing you can really hear just at the moment is the kettle."

Instead of smelling the Kitchen during Sauteeing Onions:
    try smelling the onions.

Instead of smelling the onions during Sauteeing Onions:
    say "The onions smell marvelous."

Instead of opening the oven during Preheating the Oven:
    say "It'll never heat if you open it up while it's warming."

Heating Kettle is a scene. Heating Kettle begins when the hot kettle contains cool water.

Before printing the name of the kettle when the kettle is hot:
    say "hot "

When Heating Kettle begins:
    say "The kettle begins to heat up."

Heating Kettle ends when the time since Heating Kettle began is 7 minutes.

Hearing the Kettle Whistle is a scene. Hearing the Kettle Whistle begins when Heating Kettle ends. Hearing the Kettle Whistle ends when the kettle is not hot.

When Hearing the Kettle Whistle begins:
    now the water is boiling;
    say "The kettle begins to burble and whistle shrilly."

When Hearing the Kettle Whistle ends:
    say "The kettle's screaming dies off."

Idling is a scene. Idling begins when play begins. Idling ends when Sauteeing Onions begins.

Sauteeing Onions ends in disaster when Scorching Onions begins.

Sauteeing Onions ends in success when the onions are sauteed and onions are not in a hot container.

Definition: a thing is alone if it is in a container which contains exactly one thing.

Sauteeing Onions ends in mixture when the sauteed onions are not alone.

When Sauteeing Onions ends in mixture:
    say "The mixture of things in [the holder of the onions] stops them cooking quite so fast."

When Sauteeing Onions ends in success:
    say "Nice work with the onions."

Every turn:
    if diced onions have been in a hot pan for ten turns:
        say "The onions are starting to look ready.";
        now the onions are sauteed.

Scorching Onions begins when Sauteeing Onions ends in disaster. Scorching Onions begins when the alone sauteed onions are in a hot container.

Scorching Onions ends horribly when the time since Scorching Onions began is three minutes. Scorching Onions ends in reprieve when the sauteed onions are not in a hot container. Scorching Onions ends in mixture when the sauteed onions are not alone.

When Scorching Onions ends in mixture:
    record "sauteeing onions" as achieved;
    say "The mixture of things in [the holder of the onions] stops them cooking quite so fast."

When Scorching Onions ends horribly:
    move smoke to Kitchen;

now the onions are burnt.

When Scorching Onions ends in reprieve:
    record "sauteeing onions" as achieved;
    say "You've got the onions off heat before they can scorch -- a good sign."

Instead of taking the onions when the onions are in the pan: try taking the pan.

Instead of smelling in the presence of the smoke:
    say "The scent of the late disaster lingers in the air."

Reconstituting the Apricots is a scene.

Reconstituting the Apricots begins when the dried apricots are in a container which contains boiling water.

When Reconstituting the Apricots begins:
    say "The apricots slowly begin to plump up again."

Reconstituting the Apricots ends when the dried apricots are not in a container which contains boiling water.

Every turn:
    if dried apricots have been in a container which contains boiling water for ten turns:
        say "The apricots have turned plump(ish).";
        now the apricots are reconstituted;
        record "reconstituting apricots" as achieved.

Test sautee with "peel onions / cut onions / get onions / put onions in pan / get sage / wash sage / cut sage / wait / wait / wait / wait / wait / wait / wait / get pan".

Test apricots with "get kettle / open kettle / get apricots / put apricots in kettle / put kettle on stove / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait / wait".

Test chicken with "open refrigerator / get chicken / open drawer / get mallet / hit breast".

Test cheese with "get scissors / get cheese / unwrap cheese".

Test me with "full score / test sautee / full score / test apricots / full score / test chicken / full score / test cheese / full score".

And... at that point you're a lot less close to being done than you think. The filling -- onions, sage, apricot, and cheese -- must be assembled and put in the chicken breasts; these tied up in string; each roll dipped in egg yolk and rolled in panko crumbs; these arranged in the Calphalon pan and baked. Then later, the whole retrieved from the oven, and the breasts transferred to a plate while we deglaze the pan and concoct the sauce with the chicken broth, wine, butter, etc. Then the chicken is sliced and plated, and the sauce poured over top. Usually one also wants a side dish or two. A number of things can go interestingly wrong in this process, of course, and implementing it would require, among other things, an intelligent management of all the possible mixtures that result.

ExampleRubies

A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

The trick here is that we need a table with text in order to keep track of the players' names.

Part 1 largely replicates the source from "Identity Theft"; new material starts at Part 2.

```
"Rubies"

Use scoring.

Part 1 - Collecting Names

The player's forename is a text that varies. The player's
full name is a text that varies.

When play begins:
    now the command prompt is "What is your name? >
".

To decide whether collecting names:
    if the command prompt is "What is your name? > ",
yes;
    no.

After reading a command when collecting names:
    if the number of words in the player's command is
greater than 5:
        say "[paragraph break]Who are you, a member of
the British royal family? No one has that many names.
Let's try this again.";
        reject the player's command;
    now the player's full name is the player's command;
    now the player's forename is word number 1 in the
```

```
player's command;
    now the command prompt is ">";
    say "Hi, [player's forename]!";
    say "[banner text]";
    move the player to the location;
    reject the player's command.



Instead of looking when collecting names: do nothing.



Rule for printing the banner text when collecting names:
do nothing.



Rule for constructing the status line when collecting
names: do nothing.



Part 2 - Adding the Leaderboard



File of Leaderboard is called "leaderboard".



When play begins:
    if the File of Leaderboard exists:
        read File of Leaderboard into the Table of Leaders;
        sort the Table of Leaders in reverse scored amount
order.



When play ends:
    choose row 10 in the Table of Leaders; [we've sorted
the table, so the lowest score will be the one at the
bottom]
    if the score is greater than scored amount entry:
        now name entry is the player's forename;
        now the scored amount entry is the score;
    show leaderboard;
    write the File of Leaderboard from the Table of
```

```
Leaders.



To show leaderboard:
    sort the Table of Leaders in reverse scored amount
order;
    say "Current leading scores: [paragraph break]";
    say fixed letter spacing;
    repeat through Table of Leaders:
        if scored amount entry is greater than 0:
            say " [name entry]";
            let N be 25 minus the number of characters in
name entry; [here we want to space out the scores so
they make a neat column]
            if N is less than 1, now N is 1;
            say N spaces;
            say "[scored amount entry][line break]";
    say variable letter spacing.



To say (N - a number) spaces:
    repeat with index running from 1 to N:
        say " ".



Table of Leaders

name                scored amount
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
"Smithee"           0
```

And now we introduce a scenario that allows different players to come up with different scores -- admittedly not a very interesting scenario, but it will do for now:

Part 3 - Scenario

Carry out taking something which is not handled:
    increment score.

The Big Treasure Chamber is a room. It contains a ruby, an emerald, a gold tooth, an antique katana, and a silver coin.

Instead of waiting, end the story finally.

Test me with "get ruby / z".

Example Battle of Ridgefield *WI*

Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

Occasionally, a piece of IF is sufficiently serious that it feels bathetic to offer the player the usual restore-restart-undo-quit options at the end. The following would replace "*** You have died ***" with a centered epitaph, then quit the game when the player hits a key.

This example relies on a standard extension to avoid any fancy programming:

"Battle of Ridgefield"

Include Basic Screen Effects by Emily Short.

```
Ridgefield is a room.


Instead of doing something when the turn count is
greater than 1: say "Alas, you no longer have the
strength."; end the story.


Rule for printing the player's obituary:
    say paragraph break;
    center "In defense of American Independence";
    center "at the Battle of Ridgefield, April 27, 1777,";
    center "died Eight Patriots who were laid in this
ground,";
    center "Companioned by Sixteen British Soldiers,";
    center "Living, their enemies,";
    center "Dying, their guests";
    say paragraph break;
    wait for any key;
    stop game abruptly;
    rule succeeds.
```

Not mentioning UNDO in the final set of options.

By default, Inform reminds the player that he has the option of typing UNDO after a story-ending action. This is generally good practice, especially for the sake of novice players who might not be aware of this possibility otherwise, and might be frustrated by a loss they could easily step back from.

Just occasionally, though, we may decide that the player does not deserve any such notification:

```
"Finality"


Cliff Edge is a room. "This narrow strip overlooks a
gorge many hundreds of feet deep, at whose bottom is
```

```
a river of molten lava. The walls of the gorge are lined
with poison-tipped spikes. Furthermore, the birds that
inhabit this valley spit balls of fire. Good thing you're
safe up here."
```

The Table of Final Question Options determines what options are to be given to the player after the story ends. We can change what is mentioned there by altering the entries. (The example Jamaica 1688 explains this table in more detail, and demonstrates some other things that we might do with it.)

```
When play begins:
    choose row with a final response rule of immediately
undo rule in the Table of Final Question Options;
    blank out the final question wording entry.


Instead of jumping:
    say "If you insist.";
    end the story.
```

And if we decided that we didn't want the player to be able to undo the command at all, we should add the use option

```
Use undo prevention.


Test me with "jump".
```

Example Jamaica 1688

Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

The options offered to the player at the end of the game are listed in the Table of Final Question Options, which means that we can add to them simply by continuing the table; what's more, the table gives us the opportunity to create a "final response rule", a rule that the game should follow in order to parse the player's input at this point.

So, for instance, if we wanted the player to be allowed to ask for notes about any of the rooms, characters, or objects in a historical game:

```
"Jamaica 1688"

Use scoring.

Section 1 - Procedure

Table of Final Question Options (continued)

[illegible]
[illegible] "[investigate something rule] for something or somewhere"
[illegible] [investigate something rule]

This is the investigate something rule:
    repeat through the Table of Footnotey Stuff:
        if the player's command matches the topic entry:
            say "[revelation entry][paragraph break]";
            rule succeeds;
    say "I'm afraid I have no revelation to vouchsafe
there."

Section 2 - Scenario

The Upper Deck is a room. Lucius is a man in the Upper
Deck.

The maximum score is 501.
```

When play begins: now the score is 501; end the story finally.

Table of Footnotey Stuff

topic | relation
"Lucas [is cast]" | "Lucas is based on a historical buccaneer who sailed with William Dampier. The original did carry a Greek New Testament, from which he read aloud when the men were stranded in the jungles near Panama."
"The [Upper Deck]" | "The [Upper Deck] is a simplified and tidied representation of a pirate sloop ca. 1688."

ExampleXerxes

Offering the player a menu of things to read after winning the game.

Building a menu is moderately tedious, so we will rely on the standard menu extensions provided. Thus:

"Xerxes"

Include Basic Screen Effects by Emily Short. Include Menus by Emily Short.

Table of Amusing Matter

topic | relation
"Did you [paragraph break] banning the worship of Seth? [line break] of Dionysus? [line break] assigning all your priests to Re? [line break] assigning male priests to Cybele? [line break] assigning married priestesses to Hestia? [line break] identifying one god as another (e.g., Isis and Hecate)? [line break] identifying

a mortal as a god (e.g., Alexander as Helios-Apollo)?"
"Did you try [paragraph break] allying a Greek city-state with the Persians? (try >MEDIZE) [line break] playing Athens as a land-based power?"

Rule for amusing a victorious player:
now the current menu is the Table of Amusing Matter;
now the current menu title is "Things to Try";
carry out the displaying activity;
clear the screen.

Omitting about a half million words from this rigorous and educational but nonetheless enthralling simulation of centuries of history, culture, and religion, we will skip directly to:

Athens is a room.

Use scoring.

Every turn:
if the score is greater than 10000, end the story finally.

When play begins: now the score is 10001.

Test me with "z".

ExampleLabyrinth of Ghosts
Remembering the fates of all previous explorers of the labyrinth.

A tradition among Nethack-like computer games of the old school is that a player's death in a given place leaves a ghost behind to haunt subsequent players. Information about past lives is sometimes stored in a "bones file", and in this example we do exactly that, for a grievously unfair little dungeon.

To begin with, the labyrinth itself. We create a kind of value to remember possible means of death in these tunnels, and we assign a coordinate position in some grid to each location. (We do this because grid positions can safely be stored in tables saved out to external files, whereas room names cannot - they represent data which changes each time we amend the source.)

```
"Labyrinth of Ghosts"


Use scoring.


A demise is a kind of value. The demises are drowned, buried by a rockfall, pierced by an arrow and slain. The latest demise is a demise that varies.


A grid location is a kind of value. (1,19) specifies a grid location. A room has a grid location called coordinates.


The Gateway is a room. "For the foolhardy adventurer, the perilous labyrinth lies north, east or south." The coordinates are (6,6). The Tomb is east of the Gateway. The coordinates are (7,6). The Rockfall Cave is north of the Gateway. "This partly fallen cave may perhaps extend further north." The coordinates are (6,5). Instead of going north in the Rockfall Cave, have the player buried by a rockfall. The Archery Canyon is south of the Gateway. "No telling why this canyon is named after archery, but perhaps if you wait around you'll find out." The coordinates are (6,7). Instead of waiting in the Archery Canyon, have the player pierced by an arrow. The Rock Pool is east of the Tomb. The coordinates are (8,6). The cold mountain pool is in the Rock Pool. The cold pool is fixed in place. Instead of entering the cold
```

```
mountain pool, have the player drowned.


Every turn when a random chance of 1 in 10 succeeds:
    say "A dwarf appears out of nowhere, and throws a
nasty little knife.";
    have the player slain.
```

And as compensation for these hazards:

```
Some silver bars are in the Tomb. The emerald is in the
Rock Pool. The platinum pyramid is in the Canyon.


Table of Point Values
item score

silver bars 8
platinum pyramid 10
emerald 4


Report taking an item listed in the Table of Point Values:
    increase the score by the score entry;
    blank out the whole row.
```

We are now ready for the actual undertaking. The Table of Ghostly Presences holds up to twenty death notices, and is initially blank. Deaths are sequentially numbered, and this number is stored in the sequence column.

```
Table of Ghostly Presences

sequence location
a number a room
with 19 blank rows.
```

As the story file starts up, we look to see if a ghosts file already exists. If one does, we load up the Table of Ghostly Presences with it: and if not,

as will be the case the first time the player explores, we leave the table blank. We sort the table so that it has earlier deaths (lower sequence numbers) first.

```
The File of Ghosts is called "ghosts".


When play begins:
    if the File of Ghosts exists, read File of Ghosts into
the Table of Ghostly Presences;
    sort the Table of Ghostly Presences in sequence
order.
```

How will ghosts manifest themselves? Because this is only a small example, we will simply tell the player that he senses something. If several ghosts are present in the same place, the most aggrieved (that is, the most recent) is sensed first...

```
After looking:
    repeat through the Table of Ghostly Presences in
reverse sequence order:
        if the haunted position entry is the coordinates of
the location, say "You sense the ghostly presence of an
adventurer, [manner of death entry] with a score of
[score at death entry] in [turns at death entry] turns."
```

(For instance, "You sense the ghostly presence of an adventurer, buried by a rockfall with a score of 10 in 5 turns.") That just leaves the rule for bumping off the player. When the Table is full, and there are already 20 ghosts, the one who died longest ago (with the lowest sequence count) is eliminated, and his row blanked out. (This will always be row 1 since we sorted the table in sequence order on reading it in.)

```
To have the player (sticky end - a demise):
    let the new sequence number be 0;
    repeat through the Table of Ghostly Presences:
        let S be the sequence entry;
        if S is greater than the new sequence number, let
the new sequence number be S;
    increment the new sequence number;
    if the number of blank rows in the Table of Ghostly
Presences is 0:
```

```
        choose row 1 in the Table of Ghostly Presences;
        blank out the whole row;
    choose a blank row in the Table of Ghostly
Presences;
    now the sequence entry is the new sequence
number;
    now the manner of death entry is the sticky end;
    now the turns at death entry is the turn count;
    now the score at death entry is the score;
    now the haunted position entry is the coordinates of
the location;
    write the File of Ghosts from the Table of Ghostly
Presences;
    now the latest demise is the sticky end;
    end the story saying "You have been [latest demise]".
```

Strictly speaking we ought to worry that after 2,147,483,647 deaths, the sequence numbers would grow too large to store in a single value, and then the sequence of ghosts will be erratic. But it seems unlikely that anyone will play this example 2.1 billion times.

Example **Big Sky Country** *WI*

138

Allowing the player to continue play after a fatal accident, but penalizing him by scattering his possessions around the game map.

Some older games allowed the player to be resurrected after a death, but punished him by distributing his possessions far and wide. Here we emulate that effect.

```
"Big Sky Country"

Use scoring.

When play begins: say "There's a bit of a drive over
from Anaconda, Montana, and then through a couple or
three ghost towns, but finally you find what you're
looking for, and strike out on foot..."
```

Entrance to Devil's Canyon is a room. "You are at the top of a steep road, which proceeds down into the canyon proper." A sign is in Devil's Canyon. It is fixed in place. "An ominous sign has been put up by the local sheriff's office." The description is "PROCEED AT OWN RISK - NO RESCUES!"

Instead of going down when a random chance of 1 in 3 succeeds:
    say "Whoooops, your footing is not as secure as you thought...";
    end the story.

Dusty Path is below Entrance. "A dusty path, with grey-brown thorny bushes on either side. Immediately to your right is a sheer drop; far below you can see the rusting remains of a Model T that some fool tried to drive by here."

Hairpin is below Dusty Path. "A sharp bend in the road, doubling back down towards the bottom of the canyon. Just north of here there is also a small cavern of some kind[if the stick pin is in the cavern], which attracts your eye with some glittery thing[end if]."

The Cavern is north of Hairpin. "Really not much more than a little hollow in the side of the canyon." In the cavern are a snake and a diamond stick pin. The snake is an animal. The description of the snake is "You're no expert, but it looks like a rattler."

Instead of taking the diamond stick pin in the presence of the snake: say "Turns out the snake is partial to that there pin, and takes exception to your intending to make off with it."; end the story.

In a fuller implementation of this game, we might make it possible to get by the snake, but in this version, it's just going to remain troublesome.

```
Crooked Path is below Hairpin. "You're about two thirds
of the way down to the bottom of the cavern at this
point."

At the Spot is below Crooked Path. "This'll be it: a bare
patch of ground that might as well have an X painted
right on it."

Rule for supplying a missing noun while digging:
    now noun is the location.

Understand "dig" or "dig hole/here" or "dig in ground/dirt/
earth" as digging. Digging is an action applying to one
thing.

Instead of digging at the spot:
    say "You dig and dig, and after a half hour or so, sure
enough, you do turn up a big box of gold! You're going
to be richer than God and Bill Gates put together.";
    increase the score by 5;
    end the story finally.

Instead of digging at the spot when the player does not
carry the shovel:
    say "What, without your shovel? That won't work too
well."

The player carries a walking stick. The player wears a
hat, a whistle, and a daypack. The daypack contains a
mylar blanket, a granola bar, a cellular phone, a water
bottle, a folding shovel, and a photocopied map. The
```

```
granola bar is edible. Instead of drinking the water, say
"You quench your thirst, for the time being." The
description of the map is "The map shows the winding
path of Devil's Canyon, with a large X down by the
south end. That would be where your uncle Jesse
buried the gold from the train robbery."


The maximum score is 5.


When play ends when the story has not ended finally:
    say "Oh dear, that ought to be fatal! However, if you
like I can get you out of it...


    Shall I? >";
    if the player consents:
        repeat with item running through things had by the
player:
            move the item to a random visited room;
        say "A strong wind picks you up and sets you back
at [the location], though perhaps minus a few of your
things.";
        resume the story;
        try looking.
```

"If the player consents" is just a convenient way to ask a yes/no question that the player must answer before going on with the game.

Chapter 12: Typography, Layout, and Multimedia Effects

. Typography; *§12.2. The Status Line*; *§12.3. Footnotes*; *§12.4. Timed Input*; *§12.5. Glulx Multimedia E*

Contents of *The Inform Recipe Book*
Chapter 11: Out Of World Actions and Effects
Chapter 13: Testing and Publishing
Indexes of the examples

§12.1. Typography

Story files produced by Inform tend not to contain elaborate typographical effects. They would only distract. Like a novel, a classic work of IF is best presented in an elegant but unobtrusive font. Inform does, however, provide for italic and bold-face, and also for a typewriter-style fixed pitch of lettering:

```
"This is an [italic type]italicised[roman type] word."
"This is an [bold type]emboldened[roman type] word."
"This is a [fixed letter spacing]typewritten[variable letter spacing]
word."
```

Authors making very frequent use of these might like to borrow the briefer definitions in **Chanel Version 1**.

A very wide range of letter-forms is normally available (and even more in quoted text), so that the writer seldom needs to not worry whether, say, a sentence like

```
A ticket to Tromsø via Østfold is in the Íslendingabók.
```

will work. **The Über-complète clavier** is an exhaustive test of such exotica.

Coloured type is trickier, and its availability depends on the story file format. For a Z-machine story, **Garibaldi 2** demonstrates this.

Finally, **Tilt 3** combines unusual letterforms (suit symbols) with red and black colours to render hands of cards typographically.

Start of Chapter 12: Typography, Layout, and Multimedia Effects

Back to Chapter 11: Out Of World Actions and Effects: §11.6. Ending The Story

Onward to §12.2. The Status Line

Example 69: ★ **Garibaldi 2** Adding coloured text to the example of door-status readouts.

Example 453: ★ **Chanel Version 1** Making paired italic and boldface tags like those used by HTML for web pages.

Example 450: ★★ **Tilt 3** Displaying the card suits from our deck of cards with red and black colored unicode symbols.

Example 70: ★★★ **The Über-complète clavier** This example provides a fairly stringent test of exotic lettering.

---

§12.2. The Status Line

The status line is the reverse-coloured bar along the top of the window during play, which conventionally, but not necessarily, shows the current location, the score (or sometimes the time of day) and the number of turns so far. It has been highly traditional since the early 1980s (Infocom's customer newsletter was for many years called "The Status Line"): it has become the visual identifier of IF. It plays the same role for IF that a header with chapter name and page number plays in a printed book.

The status line is ordinarily printed from two named pieces of text, the "left hand status line" and the "right hand status line". These can be changed during play, so for instance,

When play begins, change the right hand status line to "Time: [time of day]".

The examples below offer miscellaneous alternatives, and are fairly self-descriptive.

See **Viewpoint** *for a way to make the status line list the player's current identity*

Start of Chapter 12: Typography, Layout, and Multimedia Effects

Back to §12.1. Typography

Onward to §12.3. Footnotes

Example 114: **Politics as Usual** Have the status line indicate the current region of the map.

Example 350: **Ways Out** A status line that lists the available exits from the current location.

Example 387: **Blankness** Emptying the status line during the first screen of the game.

Example 412: **Capital City** To arrange that the location information normally given on the left-hand side of the status line appears in block capitals.

Example 452: **Status line with centered text, the hard way** A status line which has only the name of the location, centered.

Example 351: **Guided Tour** A status line that lists the available exits from the current location, changing the names of these exits depending on whether the room has been visited or not.

Example 115:

Centered Replacing the two-part status line with one that centers only the room name at the top of the screen.

§12.3. Footnotes

Ibid. provides a version of the traditional Infocom-style approach to footnotes, which number off in the order encountered.

Start of Chapter 12: Typography, Layout, and Multimedia Effects
Back to §12.2. The Status Line
Onward to §12.4. Timed Input

Example 300: ★★ **Ibid.** A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

§12.4. Timed Input

Inform normally expects a purely turn-based story: the player acts, the story responds and waits for the player to act again.

Occasionally, however, we may want to offer a different mode of interaction, for instance with turns in which the player has limited time to come up with his next act. Likewise, we might want to have text that printed itself to the screen gradually, to represent dialogue with pauses, or the speed of a typewriter placing letters on a page.

It's best to be careful with these effects: overdone, they can be very annoying to players who prefer to read at a faster speed. Nonetheless, they do have their uses.

Inform does not have standard syntax to handle real-time delays and output, but there are several extensions that provide this capacity. Erik Temple's extension Real Time Delays, for instance, allows us to specify a delay in milliseconds before continuing with whatever aspect of the story is currently in progress.

See **The Passage Of Time** *for ways to keep track of clock-time within the story*

Start of Chapter 12: Typography, Layout, and Multimedia Effects
Back to §12.3. Footnotes
Onward to §12.5. Glulx Multimedia Effects

§12.5. Glulx Multimedia Effects

Glulx is one of the two basic story file formats to which Inform can work. It is the more powerful of the two, and modern-day Inform uses it by default. At one time it was a less universally playable format, but today players rarely have any trouble getting it to work.

Among its powers are the ability to display images, play back sound effects, and read and write external files to the disc. With care and a certain amount of fuss, this can even give a playing story file limited Internet connectivity, although it should be stressed that this can only be done if the player sets up his computer just right and runs an auxiliary program beside the story itself. That will mostly be too much to ask, if the player is playing offline, but when the story file is being run on an interpreter running at a server - so that the player simply sends commands to it and sees responses back on a web page - one could easily imagine setting up the server to provide these auxiliary programs, without any extra difficulty for the player.

Many of the more advanced multimedia abilities of Glulx are best unlocked using extensions available from the Inform website or the Public Library. As of this writing, extensions exist to help authors create complex multi-windowed displays (including per-location pictures, visual status bars, and even limited animations and gradually-revealed maps).

There is also work in progress on sound-management to allow the author to play sounds in multiple channels at once, to change sound volumes, and to create fade-in and fade-out effects.

Without extensions, all these abilities are within reach for an author who is willing to do some fairly advanced programming.

Start of Chapter 12: Typography,

Layout, and Multimedia Effects
Back to §12.4. Timed Input

Onward to Chapter 13: Testing and Publishing: §13.1. Testing

Example 445: ★★★ **Flathead News Network** Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

Examples from Chapter 12: Typography, Layout, and Multimedia Effects

Start of this chapter

Chapter 13: Testing and Publishing
Indexes of the examples

★ **ExampleGaribaldi 2**
Adding coloured text to the example of door-status readouts.

WI

The extension "Basic Screen Effects" provides a few more type styles, in the form of coloured lettering. The colours available are red, yellow, green, blue, white, magenta, and cyan, as well as the usual black; and to restore the player's default screen colour, we say "default letters".

Thus if we wanted to highlight locked and unlocked doors in our security readout example:

```
"Garibaldi"


Include Basic Screen Effects by Emily Short.
```

```
The security readout is a device. The description of the
readout is "The screen is blank."



Instead of examining the switched on security readout:
    say "The screen reads: [fixed letter spacing]";
    say line break;
    repeat with item running through doors:
        say line break;
        say " [item] ([front side of the item]/[back side of the
item]): [if the item is locked][green
letters]LOCKED[default letters][otherwise][red
letters]UNLOCKED[default letters][end if]";
    say variable letter spacing;
    say paragraph break.



The player carries the security readout.



The Docking Bay is a room. The inner airlock is a door.
It is north of the Docking Bay and south of the Zocalo.
The inner airlock is lockable and unlocked. The outer
airlock is lockable and locked. It is a door. It is south of
the Docking Bay and north of Space.



The quarantine seal is a door. It is west of the Zocalo
and east of Medlab. Quarantine seal is locked.



The security pass unlocks the inner airlock. The player
carries the security pass.



Test me with "x readout / turn on readout / x readout /
lock inner airlock with security pass / x readout".
```

Note that this extension does not currently produce the desired effects when compiling with the Glulx setting; to see it working, make sure that

the settings tab is set to compile to the Z-machine.

ExampleChanel Version 1

Making paired italic and boldface tags like those used by HTML for web pages.

HTML uses angled brackets to achieve effects, and places italicised text between and tags; and similarly boldface between and . We can mimic this very easily by setting each up as a segmented substitution:

```
"Chanel Version 1"

To say i -- beginning say_i -- running on: (- style
underline; -).
To say /i -- ending say_i -- running on: (- style roman; -).
To say b -- beginning say_b -- running on: (- style bold;
-).
To say /b -- ending say_b -- running on: (- style roman;
-).

Place Vendôme is a room. "[i]Fashion fades, only style
remains the same[/i] ([b]Coco Chanel[/b]). And this
elegant drawing-room, once a milliner's shop, is a case
in point."

Instead of going nowhere, say "[i]Don't spend time
beating on a wall, hoping to transform it into a door.[/i]
([b]Coco Chanel[/b]) This one is a wall.".

Test me with "look / e".
```

We have had to use square instead of angle brackets, but then, "in order to be irreplaceable one must always be different" (Coco Chanel).

(Marking these as substitutions which run on prevents unexpected paragraph breaks if they should appear immediately after the end of a sentence.)

ExampleTilt 3

WI

Displaying the card suits from our deck of cards with red and black colored unicode symbols.

Books and articles about card-playing traditionally abbreviate card names into a simple two-symbol notation: a number or letter representing the card rank, followed by a symbol indicating the card suit. Suppose that we want to emulate this notation when taking inventory in our poker game.

The trick here is that colored output is done in different ways by the Z-Machine and by Glulx, so we'll need two different versions of the same section in order to produce this output. The relevant source is right at the beginning:

```
"Tilt"

Section 0 - Colored Output in Two Forms
```

For the suit symbols, we'll want the Unicode extension included with Inform:

```
Include Unicode Character Names by Graham Nelson.

Rule for printing the name of a card (called target) while
grouping together:
    say "[rank of the target as abbreviated value][suit of
the target as symbol]".

To say (current suit - a suit) as symbol:
    if current suit is diamonds, say "[red letters][unicode
```

```
black diamond suit][default letters]";
    if current suit is spades, say "[unicode black spade
suit]";
    if current suit is clubs, say "[unicode black club suit]";
    if current suit is hearts, say "[red letters][unicode
black heart suit][default letters]".
```

Section 0Z (for Z-machine only)

The Basic Screen Effects extension bundled with Inform includes mechanisms to change the text color, so for the Z-machine, we need only include this:

```
Include Basic Screen Effects by Emily Short.
```

Section 0G (for Glulx only)

Under Glulx, we need slightly more set-up: Glulx requires that we define special user font styles when we plan to make display changes. A fuller discussion of this (and of how to define new colors) appears in the documentation of "Glulx Text Effects", but an implementation sufficient to our purposes would be

```
Include Glulx Text Effects by Emily Short.
```

Table of User Styles (continued)

style name	style value
special-style-1	special-style

Table of Common Color Values (continued)

color name	color value
gold	16711680

```
To say red letters: say first custom style.

To say default letters: say roman type.
```

From here, the rest of the source is mostly as we've seen in previous examples:

```
Section 1 - Cards

Suit is a kind of value. The suits are hearts, clubs,
diamonds, and spades. Understand "heart" as hearts.
Understand "club" as clubs. Understand "diamond" as
diamonds. Understand "spade" as spades.

A card is a kind of thing. A card has a suit. A card has a
number called rank. Understand the suit property as
describing a card. Understand the rank property as
describing a card.

52 cards are in the card repository.

To say (count - a number) as a card value:
    choose row count in the Table of Value Names;
    say "[term entry]".

Rule for printing the name of a card (called target):
    say "[rank of the target as a card value] of [suit of the
target]"
```

To say (count - a number) as abbreviated value:
    choose row count in the Table of Value Names;
    say "[abbrev entry]".

Table of Value Names

table	abbrev
"ace"	"A"
"deuce"	"two"
"three"	"3"
"four"	"4"
"five"	"5"
"six"	"6"
"seven"	"7"
"eight"	"8"
"nine"	"9"
"ten"	"10"
"jack"	"knave/J"
"queen"	"Q"
"king"	"K"

After reading a command:
    if the player's command includes "of [suit]":
        while the player's command includes "of":
            cut the matched text;
    repeat through the Table of Value Names:
        while the player's command includes topic entry:
            replace the matched text with value entry.

When play begins:
    reconstitute deck.

To reconstitute deck:
    let current suit be hearts;
    now every card is in the card repository;
    while a card is in the card repository:
        repeat with current rank running from 1 to 13:
            let item be a random card in card repository;
            now rank of item is current rank;
            now suit of item is current suit;

now item is in the deck of cards;
now current suit is the suit after the current suit.

Section 2 - The Deck and the Discard Pile

The Empty Room is a room. "Nothing to see here."

The deck of cards is in the Empty Room. It is a closed unopenable container. The description is "A standard poker deck."

The discard pile is a closed unopenable container. The description is "Cards in this game are discarded face-down, so the discard pile is not very interesting to see. All you can observe is that it currently contains [if the number of cards which are in the discard pile is less than ten][the number of cards which are in the discard pile in words][otherwise]about [the rounded number of cards which are in the discard pile in words][end if] card[s]."

To decide what number is the rounded number of (described set - a description of objects):
    let N be the number of members of the described set;
    let R be N divided by 5;
    let total be R times 5;
    decide on total.

Rule for printing room description details of something: do nothing instead.

Section 3 - Drawing and Discarding Actions

Understand the commands "take" and "carry" and "hold" and "get" and "drop" and "throw" and "discard" as something new.

Understand "take [text]" or "get [text]" or "drop [text]" as a mistake ("Here, you only draw and discard. Nothing else matters at the moment.").

Understand "draw" or "draw card" or "draw a card" as drawing. Drawing is an action applying to nothing. The drawing action has an object called the card drawn.

Setting action variables for drawing:
    now the card drawn is a random card which is in the deck of cards.

Check drawing:
    if the card drawn is nothing, say "The deck is completely depleted." instead.

Check drawing:
    if the number of cards carried by the player is greater than four,
        say "This is a five-card game; you must discard something before drawing anything further." instead.

Carry out drawing:
    move the card drawn to the player.

Report drawing:
    say "You draw [a card drawn]."

```
Understand "discard [card]" as discarding. Discarding is
an action applying to one thing.


Check discarding:
    if the player does not carry the noun, say "You can
only discard cards from your own hand." instead.


Carry out discarding:
    now the noun is in the discard pile;
    if the discard pile is not visible, move the discard pile
to the location.


Report discarding:
    say "You toss [the noun] nonchalantly onto the
discard pile."


Section 4 - Assessing Hands


Before listing contents while taking inventory: group
cards together.


Before grouping together cards:
    if the number of cards carried by the player is 5:
        say "[run paragraph on]";
        follow the hand-ranking rules;
        if the rule succeeded, say "[the outcome of the
rulebook]";
        otherwise say "some random cards";
        if the outcome of the rulebook is pair outcome, say
" of [rank of the first thing held by the player as a card
value]s";
    otherwise:
        say "[number of cards carried by the player in
words] assorted cards";
    say " (".
```

```
Rule for grouping together cards:
    say "[list hand]".



To say list hand:
    let chosen card be the first thing held by the player;
    while chosen card is a card:
        say "[chosen card]";
        now chosen card is the next thing held after
chosen card;
        if chosen card is a card, say "-".



After grouping together cards:
    say ")".
```

The hand-ranking rules is a rulebook. The hand-ranking rules have outcomes royal flush, straight flush, four of a kind, full house, flush, straight, three of a kind, two pairs, pair, high card.

The hand-ranking rulebook has a truth state called the flushness.
The hand-ranking rulebook has a truth state called the straightness.

The hand-ranking rulebook has a number called the pair count.
The hand-ranking rulebook has a number called the triple count.
The hand-ranking rulebook has a number called the quadruple count.

A card can be sorted or unsorted. A card is usually unsorted.

Definition: a card is high if its rank is 11 or more.
Definition: a card is low if its rank is 4 or less.

A hand-ranking rule (this is the initial sort rule):
    now every card is unsorted;
    while the player carries an unsorted card:
        let item be the lowest unsorted card held by the player;
        move item to the player;
        now the item is sorted;
    if sort-debugging is true, say "-- after initial sort: [list hand]".

A hand-ranking rule (this is the finding flushness rule):
    let called suit be the suit of a random card carried by the player;
    if every card carried by the player is called suit, now flushness is true.

A hand-ranking rule (this is the finding straightness rule):
    now straightness is true;
    let N be the rank of the highest card which is carried by the player;
    repeat with current rank running from N - 4 to N:
        now the test rank is the current rank;
        unless the player carries a matching card:
            if the current rank is N - 4 and the current rank is 9 and the player carries an ace card, do nothing;
            otherwise now straightness is false.

A card can be quadrupled, tripled, paired or uncombined.

Test rank is a number that varies. Definition: a card is matching if its rank is the test rank.

```
A hand-ranking rule (this is the counting multiples rule):
    now every card is uncombined;
    repeat with current rank running from 1 to 13:
        now test rank is current rank;
        let N be the number of matching cards held by the
player;
        if N is 4:
            increment the quadruple count;
            now every matching card held by the player is
quadrupled;
        if N is 3:
            increment the triple count;
            now every matching card held by the player is
tripled;
        if N is 2:
            increment the pair count;
            now every matching card held by the player is
paired.



A hand-ranking rule (this is the move aces up unless
there's a low straight rule):
    unless the straightness is true and the lowest card
carried by the player is an ace card and the rank of the
highest card carried by the player is 5,
        now every ace card which is carried by the player
is carried by the player;
    if sort-debugging is true, say "-- after ace movement
rule: [list hand]".



A hand-ranking rule (this is the move pairs forward rule):
    while the player carries a paired card:
        let selection be the lowest paired card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after pairs movement:
[list hand]".



A hand-ranking rule (this is the raise ace pairs rule):
    if the player carries exactly two ace cards:
        repeat with item running through ace cards which
```

```
are carried by the player:
            move item to the player;
    if sort-debugging is true, say "-- after paired-ace
movement: [list hand]".



A hand-ranking rule (this is the move multiples forward
rule):
    while the player carries a tripled card:
        let selection be the lowest tripled card which is
carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    while the player carries a quadrupled card:
        let selection be the lowest quadrupled card which
is carried by the player;
        move the selection to the player;
        now the selection is uncombined;
    if sort-debugging is true, say "-- after multiples
movement rule: [list hand]".



Definition: a card is ace if its rank is 1.
Definition: a card is king if its rank is 13.



A hand-ranking rule (this is the royal-flush rule):
    if flushness is true and straightness is true and the
highest card carried by the player is king and the lowest
card carried by the player is ace, royal flush.



A hand-ranking rule (this is the straight-flushes rule):
    if flushness is true and straightness is true, straight
flush.



A hand-ranking rule (this is the four-of-a-kind rule):
    if the quadruple count is 1, four of a kind.



A hand-ranking rule (this is the full-house rule):
```

if the pair count is 1 and the triple count is 1, full house.

A hand-ranking rule (this is the flushes rule):
if flushness is true, flush.

A hand-ranking rule (this is the straights rule):
if straightness is true, straight.

A hand-ranking rule (this is the three-of-a-kind rule):
if triple count is 1, three of a kind.

A hand-ranking rule (this is the two-pair rule):
if the pair count is 2, two pairs.

A hand-ranking rule (this is the pair rule):
if the pair count is 1, pair.

A hand-ranking rule (this is the default rule):
high card.

Sort-debugging is a truth state that varies.

Section 5 - Testing hand identification - Not for release

Understand "debug sorting" as debugging hand sorting.
Debugging hand sorting is an action out of world.

Carry out debugging hand sorting:
    if sort-debugging is false, now sort-debugging is true;
    otherwise now sort-debugging is false.

Report debugging hand sorting:
    say "Sort debugging is now [if sort-debugging is true]on[otherwise]off[end if]."

Test me with "draw / g / g / g / g / force hand / g / g / g / g / g / g / g / g / g / g / g".

Table of Testing Hands

set card
[royal flush]
spades 13
spades 12
spades 11
spades 10
[straight flush]
clubs 11
clubs 10
clubs 9
clubs 8
[four of a kind]
hearts 8
spades 8
clubs 8
clubs 3
[full house]
spades 4
hearts 10
spades 10
clubs 10
[flush]
hearts 5
hearts 7
hearts 11
hearts 12
[straight]
spades 13
diamonds 12
clubs 11
hearts 10

[three of a kind]
8 spades
2 clubs
4 clubs
3 spades

[two pairs]
6 spades
7 clubs
7 diamonds
9 hearts

[two pairs, ace high]
6 spades
1 clubs
7 diamonds
1 hearts

[pair]
12 spades
10 diamonds
7 spades
4 clubs

[high]
13 diamonds
11 hearts
9 spades
7 clubs
5 diamonds

[tricky sorting: low straight]
2 diamonds
3 spades
4 diamonds
5 diamonds

Understand "force hand" as forcing a hand. Forcing a hand is an action out of world.

Current marker is a number that varies.

Carry out forcing a hand:
    repeat with item running through cards which are carried by the player:
        increment current marker;
        if current marker is greater than the number of filled rows in the Table of Testing Hands, now current marker is 1;

choose row current marker in the Table of Testing Hands;
now the suit of item is the set suit entry;
now the rank of item is the set rank entry.

Report forcing a hand:
try taking inventory.

ExampleThe Über-complète clavier *WI*

This example provides a fairly stringent test of exotic lettering.

The following example puts Inform's support for exotic lettering through its paces. It was useful in testing Inform but is not a very instructive read: still, it does provide a test story file for interpreters, so we are including the source here as an example.

"The Über-complète clavier"

The story headline is "Pushing the Limits of Unicode in IF". The story description is "This is a demanding test for Unicode compliance by Z-machine interpreters."

Include Unicode Character Names by Graham Nelson.

Include Basic Screen Effects by Emily Short.

The Château Bibliothèque Français is east of the Deutsche Universität Bücherei. "From this Borgesian construction, doorways lead into anterooms in each of the four cardinal directions." South of the Bibliothèque is the Miscellany Mañana. North of the Bibliothèque is the

Íslendingabók. East of the Bibliothèque is Alphabet Soup.

A framed photograph of Icelandic Prime Minister Halldór Ásgrímsson, a ruler measuring Ångströms, a Bokmål-Lëtzebuergesch Lëtzebuergesch-Bokmål dictionary and a ticket to Tromsø via Østfold are in the Íslendingabók.

A paper by Karl Weierstraß, a general feeling of Ärger, an old Österreich passport and the Bach Clavier-Übung open at the fugue à 4 are in the Bücherei.

The painting of École normale superiéure students singing Ça ira, the frankly lesser-known journal of Niccolò Polo, Così fan tutte on CD, an extract of Herodotus concerning Artaÿctes and the exit sign reading À BIENTÔT are in the Bibliothèque.

A wicker basket marked CHLOË is in the Bibliothèque. A ginger cat is in the basket.

A guide to Æsop for naïve æsthetes, Lönnrot's Kalevala, a creed according to the Bahá'ís, FALARÃO magazine, an Estonian poem by Tõnu Trubetsky, a Portuguese-Italian recipe for macarrão, a stripy hanging CANDY PIÑATA bag, a ¿¡Punctuation Turned Upside Down¿¡ pamphlet, an Italian brewers' anti-violence poster declaring BÓTTE NON BÒTTE, a map of È and a dusty book titled The Parnasum of Luís Vaz bearing CAMÕES on its spine are in Miscellany Mañana.

The description of the map is "È is a province in the People's Republic of China."

In Mañana is something called ÂÊÎÔÛ - The Official Journal of the Society for Vowels bearing Circumflexes.

In Mañana is something called âêîôû comic - the youth edition.

The description of Alphabet Soup is "A bewildering place of glyphs, sigils and signs. The Library proper leads back west: steps lead upwards to an Observatory, or downwards into what seems to be a dangerous area. A gaming lounge lies to the south."

The Greek Alphabet, the Cyrillic Alphabet, the Hebrew alphabet, and the embossed plaque are in Alphabet Soup. The description of the Greek alphabet is "αβγδεζηθικλμνξοπρςστυφχψω.". The description of the Hebrew alphabet is "אבגדהוזחטיךכלמנסעף פץצקרשת.". The description of the Cyrillic alphabet is "абвгдежзийклмнопрстуфхцчшщъыьэюя.".

Instead of examining the plaque:
    say "It seems to be a sign in Braille: ";
    say unicode Braille pattern dots-24, " (I), ",
        unicode Braille pattern dots-1345, " (N), ",
        unicode Braille pattern dots-124, " (F), ",
        unicode Braille pattern dots-135, " (O), ",
        unicode Braille pattern dots-1235, " (R), ",
        unicode Braille pattern dots-134, " (M)."

The Gaming Lounge is south of Alphabet Soup. The chess position and the book of puzzle canons are in the Gaming Lounge.

The Georges de la Tour painting Le Tricheur is in the Gaming Lounge. "Hanging on one wall is Georges de la Tour's masterpiece Le Tricheur (the card-sharp). Visible

are 8[unicode black diamond suit], 9[unicode black diamond suit], A[unicode black diamond suit], A[unicode black spade suit], 6[unicode black club suit] but not one of them has a [unicode black heart suit]."

The description of Le Tricheur is "If they'd been dice-players instead, they might have thrown [unicode die face-1], [unicode die face-2], [unicode die face-3], [unicode die face-4], [unicode die face-5] or [unicode die face-6], but as it is they stick to cards."

The description of the book of canons is "A typical fugue is no. 13 (Tovey: [unicode eighth note] = 110) in F[unicode music sharp sign] minor, but you can also make out keys like A[unicode music flat sign] and G[unicode music natural sign]."

The empty square text is text that varies. To say empty: say the empty square text.

To display the board:
    say empty, empty, empty, empty, empty, empty, unicode black chess king, empty, line break;
    say empty, empty, empty, unicode black chess queen, empty, empty, unicode black chess pawn, empty, line break;
    say unicode black chess pawn, empty, empty, unicode black chess bishop, unicode black chess pawn, empty, empty, unicode black chess pawn, line break;
    say empty, empty, empty, unicode black chess pawn, empty, unicode black chess rook, empty, empty, line break;
    say empty, unicode black chess pawn, empty, unicode white chess pawn, unicode black chess pawn, empty, empty, empty, line break;
    say empty, empty, empty, unicode black chess bishop, unicode white chess queen, empty, unicode white chess pawn, unicode white chess pawn, line break;
    say unicode white chess pawn, unicode white chess pawn, empty, unicode white chess bishop, empty,

unicode black chess rook, unicode white chess bishop, empty, line break;
    say empty, unicode white chess knight, empty, empty, unicode white chess rook, empty, unicode white chess rook, unicode white chess king, line break.

Instead of examining the chess position:
    say "Fritz Saemisch - Aron Nimzowitsch, Copenhagen 1923: the Immortal Zugzwang Game. Nimzowitsch (black), observing that white will very soon have to play a terrible move, has just advanced his h pawn for no reason other than to wait. So it is white to play...";
    say "[fixed letter spacing]......k. [line break]...q..p. [line break]p..bp..p [line break]...p.r.. [line break].p.Pp... [line break]...bQ.PP [line break]PP.B.rB. [line break].N..R.RK [variable letter spacing][line break]";
    say "'White must, willy-nilly, eventually throw himself upon the sword', in Nimzowitsch's commentary. ";
    say "We will now try to display the same position using chess-piece symbols in a Unicode font.";
    say fixed letter spacing;
    now the empty square text is " ";
    display the board;
    say variable letter spacing.

The Astrological Observatory is above Alphabet Soup.

The planets are in the Observatory. "Diagrams of the planets are scattered across the dome: Sun [unicode Sun], Mercury [unicode Mercury], Venus [unicode Female Sign], Earth [unicode Earth], Moon [unicode First Quarter Moon] and [unicode Last Quarter Moon], Mars [unicode Male Sign], Jupiter [unicode Jupiter], Saturn [unicode Saturn], Uranus [unicode Uranus], Neptune [unicode Neptune], Pluto [unicode Pluto] and one or two comets [unicode Comet]. Fainter, but all around, you see stars black [unicode black star] and white [unicode white star]."

The constellations are in the Observatory. "Ringing the

dome are the constellations: Aries [unicode Aries], Taurus [unicode Taurus], Gemini [unicode Gemini], Cancer [unicode Cancer], Leo [unicode Leo], Virgo [unicode Virgo], Libra [unicode Libra], Scorpius [unicode Scorpius], Sagittarius [unicode Sagittarius], Capricorn [unicode Capricorn], Aquarius [unicode Aquarius], Pisces [unicode Pisces]."

The weather almanac is in the Observatory. The description of the almanac is "Here nightly observers scrawl in hasty abbreviations for the current weather conditions: clear weather [unicode Black Sun with Rays], cloudy [unicode cloud], rain [unicode umbrella], snow [unicode snowman], lightning [unicode lightning], thunderstorm [unicode thunderstorm]."

The Danger Zone is below Alphabet Soup. The printed name of the Danger Zone is "[unicode skull and crossbones] Danger Zone [unicode skull and crossbones]".

The warning signs are in the Danger Zone. "A variety of international-standard warning standards suggest that this may not be the safest place: [unicode skull and crossbones], [unicode caution sign], [unicode radioactive sign], [unicode biohazard sign]."

This example text was used to produce a story file which has been tried against both Zoom for Mac OS X and Windows Frotz. The Latin, Greek, Cyrillic and Hebrew text all functioned perfectly on both, but a point of difference showed when writing the Hebrew alphabet: Zoom wrote this right-to-left, Windows Frotz left-to-right. The exotic symbols displayed on Zoom (though others not mentioned above, such as "[unicode staff of hermes]", did not): but most appeared only as black squares on Windows Frotz, exceptions being the astrological signs for Venus and Mars and the musical note.

Example **Politics as Usual**

Have the status line indicate the current region of the map.

Suppose a game with a large map entirely subdivided into regions. We could define:

```
"Politics as Usual"

When play begins:
    now the right hand status line is "[map region of the
location]".

Washington is west of Idaho.

Red is a region. Blue is a region. Idaho is in red.
Washington is in blue.

Test me with "e / w".
```

Note that, since regions can be stacked, we technically can be within more than one region at once. In the Port Royal example, for instance, the Tavern region is inside the Inland region. If there is any ambiguity, "the map region of the location" will be construed as "the smallest region that the location belongs to": so we would see "Tavern" rather than "Inland" in the status bar, when the player was in the Feathers or the Feathers Bedroom.

Some extra finesse would be necessary if the names of map regions were very long or if there were some rooms that were not considered to belong to any region at all.

ExampleWays Out
A status line that lists the available exits from the current location.

A not-uncommon device in games with large maps is a list of available

exits printed in the status bar. We might do this so:

```
"Ways Out"

When play begins:
    now left hand status line is "Exits: [exit list]";
    now right hand status line is "[location]".

To say exit list:
    let place be location;
    repeat with way running through directions:
        let place be the room way from the location;
        if place is a room, say " [way]".
```

We may find that printing out full directions makes the status line unpleasantly crowded. Fortunately, it isn't hard to provide a set of abbreviations to use in this context:

```
Rule for printing the name of a direction (called the way)
while constructing the status line:
    choose row with a heading of the way in the Table of
Abbreviation;
    say "[shortcut entry]".

Table of Abbreviation

heading                shortcut entry
north                  "N"
northeast              "NE"
northwest              "NW"
east                   "E"
southeast              "SE"
south                  "S"
southwest              "SW"
west                   "W"
up                     "U"
down                   "D"
inside                 "IN"
outside                "OUT"
```

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "w / n / e / e / s / e".
```

Everywhere else, the names of directions will still be printed out in full in the usual way.

ExampleBlankness

Emptying the status line during the first screen of the game.

Occasionally we want to print something as our first screen and then pause the game. By default, Inform will print a rather odd status line, with "You" on the left side and "0" on the right. This is because the left hand status line is set to display the location, but (because we're not done with the when-play-begins rules) the player has not yet even been moved to a room.

We can tidy this up in the "starting the virtual machine" activity, by temporarily changing the status line content. We will not provide game-pausing code here, because that is easily done by extension; so:

```
"Blankness"
```

```
Include Basic Screen Effects by Emily Short.


When play begins:
    say "take me home";
    wait for any key;
    say " yeah";
    wait for any key;
    say " yeah";
    pause the game;
    now the left hand status line is "[location]";
    now the right hand status line is "[turn count]".


Before starting the virtual machine:
    now the left hand status line is "";
    now the right hand status line is "".


Paradise City is a room. The description of Paradise
City is "The grass is green and the girls are pretty."
```

Quite a modest effect, but occasionally useful.

---

412 Example Capital City *WI*

To arrange that the location information normally given on the left-hand side of the status line appears in block capitals.

Not much is needed for this. The only noteworthy point is that it doesn't work by changing the LHSL to "[the player's surroundings in upper case]": it cannot do this because "the player's surroundings" is not a value. Instead, "[the player's surroundings]" is a text substitution sometimes printing the name of a room, sometimes printing "Darkness", and so on. We must therefore load it into a text first, and then apply "...in upper case".

```
"Capital City"
```

```
Capital City is a room. East is Lower Caissons. South of
Lower Caissons
is San Seriffe. East of San Seriffe is a dark room.


To say the player's capitalised surroundings:
    let the masthead be "[the player's surroundings]" in
upper case;
    say the masthead.


When play begins:
    now the left hand status line is "[the player's
capitalised surroundings]".


Test me with "e / s / e".
```

---

452 **Example** **Status line with centered text, the hard way** *WI*

A status line which has only the name of the location, centered.

Making major changes to display features, such as the construction of the status line, sometimes requires that we rely on Inform 6 in bulk; here is how we might produce the Trinity-style status line, with the location centered at the top center of the screen.

```
"Corner of No and Where"


No is a room. Where is west of No.


Rule for constructing the status line:
```

```
    print the location in the center of the status line;
    rule succeeds.



To print the location in the center of the status line:
    (- PrintCenteredStatus(); -).



Include (-



Array printed_text --> 64;



[ PrintCenteredStatus i j;
    @set_cursor 1 0;
    i = 0->33;
    spaces(i);
    @output_stream 3 printed_text;
    print (name) location;
    @output_stream -3;
    j = (i - (printed_text-->0))/2;
    @set_cursor 1 j;
    print (name) location;
    spaces j-1;
];



-)



Test me with "w / e".
```

In fact, as we've already seen, many extra modifications to the display behavior are possible using Basic Screen Effects.

A status line that lists the available exits from the current location, changing the names of

these exits depending on whether the room has been visited or not.

It may sometimes be helpful to prompt the player with a list of exits printed up in the status line. For instance, here is a status line that will print the names of nearby rooms, as well as all the doors the player can see:

```
"Guided Tour"


When play begins:
    now left hand status line is "Nearby: [if a room is
adjacent][the list of adjacent rooms][end if][if a room is
adjacent and a door is visible] and [end if][if a door is
visible][the list of visible doors][end if]";
    now right hand status line is "".
```

Of course, we may not want to tell the player what glories are to be found in locations he hasn't yet explored.

```
Rule for printing the name of an unvisited room (called
the target) while constructing the status line:
    let aim be the best route from the location to the
target;
    say "something [aim]".
```

Even when we have seen a room, we might still want a reminder about how to get there:

```
After printing the name of a visited room (called the
target) while constructing the status line:
    let aim be the best route from the location to the
target;
    say " ([aim])".
```

We may also find that printing out full directions makes the status line unpleasantly crowded. Fortunately, it isn't hard to provide a set of abbreviations to use in this context:

```
Rule for printing the name of a direction (called the aim)
while constructing the status line:
    choose row with a heading of the aim in the Table of
Abbreviation;
    say "[shortcut entry]".
```

Table of Abbreviation

heading	shortcut entry
"N"	north
"NE"	northeast
"NW"	northwest
"E"	east
"SE"	southeast
"S"	south
"SW"	southwest
"W"	west
"up"	up
"down"	down
"inside"	inside
"outside"	outside

Everywhere else, the names of directions will still be printed out in full in the usual way. And now we give it a little map to work with:

```
Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.


The church door is east of Eastern End and west of the
Courtyard. The church door is a door.


Test me with "n / w / s".
```

Note that while this looks fine in some places, other locations exceed the limits of what the status-line can hold: if any given room is going to have a large number of exits, this kind of listing will almost certainly not fit. So apply cautiously.

ExampleCentered

Replacing the two-part status line with one that centers only the room name at the top of the screen.

If we want to lay out the status line in some other way than with left-hand and right-hand entries, it is possible to do this as well. Later we will learn about the "rule for constructing the status line", but here is a basic effect using this rule and an Inform extension included as part of the standard distribution, called Basic Screen Effects.

```
"Centered"

When play begins:
    say "After months of boring through the Earth's crust
in this metal-jawed vehicle, you break through..."

The Hollow Core is a room. "Truly a magnificent sight:
the land curves up away from you in every direction,
covered with the cities and fields of the Core People.
Molten rock runs in the canals, bringing heat and light to
every home.

At the center of the Earth hangs a dense black sun."

Include Basic Screen Effects by Emily Short.
```

```
Rule for constructing the status line:
    center "[location]" at row 1;
    rule succeeds.


Test me with "look".
```

Basic Screen Effects also provides a mechanism for building complicated status lines of more than one row. To read its documentation, we include the extension, press Go!, and then consult the contents index that results.

---

300 **Example** **Ibid.** *WI*

A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

"Hitchhiker's Guide to the Galaxy" introduced the idea of footnoted descriptions, and various IF games since have toyed with the idea. The recommended implementation in Inform 6 involved keeping an assortment of footnote objects around, but in Inform 7 the table is a much tidier way of handling the same problem.

```
"Ibid."


The Ship Inn is a room. "Here you are in a lovely pub
which your guidebook assures you is extremely
authentic. [1 as a footnote].


To your left sits a party of Italians, with their guidebook.


To your right is a silent, but not unappealing, young
man.".
```

A party of Italians and a silent young man are people in the Ship Inn. The Italians and the young man are scenery.

The table is a supporter in the Ship Inn. On the table is a mysterious pie. The description of the pie is "Your waitress told you it was the specialty of the day, Steak and Owl Pie. [2 as a footnote]." The pie is edible.

Table of Footnotes

assignment	note
1	"Francis Drake ate here, if the sign on the door is to be believed"
2	"this is unlikely, considering that owls are protected animals in England these days [3 as a footnote]"
3	"moreover, you can't imagine that owl would be very tasty"

Footnotes mentioned is a number that varies.

Whenever we mention a footnote for the first time, we need to assign it a number, which we will use consistently thereafter. And it's probably a good idea to protect ourselves against the author accidentally using a number too large for the footnote table, too. So:

```
To say (footnote - a number) as a footnote:
    if footnote > number of filled rows in the Table of
Footnotes:
        say "Programming error: footnote assignment out
of range.";
    otherwise:
        choose row footnote in the Table of Footnotes;
        if there is an assignment entry:
            say "([assignment entry])";
        otherwise:
            increment footnotes mentioned;
            choose row footnote in the Table of Footnotes;
            now assignment entry is footnotes mentioned;
            say "([assignment entry])".
```

Now, in order to let the player view these footnotes, we'll need to parse numbers.

```
Understand "footnote [number]" as looking up a
footnote.

Looking up a footnote is an action applying to one
number.

Check looking up a footnote:
    if the number understood > footnotes mentioned, say
"You haven't seen any such footnote." instead;
    if the number understood < 1, say "Footnotes are
numbered from 1."

Carry out looking up a footnote:
    choose row with assignment of number understood in
the Table of Footnotes;
    say "([assignment entry]): [note entry]."

Test me with "footnote 1 / examine pie / footnote 2 /
footnote 3".
```

This method does require us to keep track of where a footnote appears in the table. If we found this inconvenient, we could add a column to the footnote table so that we could invoke it with tags like "[appearance quip as a footnote]".

ExampleFlathead News Network

Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

This example can only work if we have a separate program running in the background while the story file is being played, and as such it will only

work if we set things up in a special way. Exactly how to do this will vary from platform to platform.

First, the source text for the Inform end of the communication line:

```
"Flathead News Network"


The file of RSS Requests is called "rssrequest".


The file of RSS Responses (owned by project "RSS-
SCRIPT") is called "rssreply".


To request (RSS feed address - text):
    mark the file of RSS Responses as not ready to read;
    write the RSS feed address to the file of RSS
Requests.


Newsroom is a room. "This is the secret nerve-centre of
FNN, the Flathead News Network."


The BBC button is in the Newsroom. Instead of pushing
the BBC button: say "Bong!"; request
"newsrss.bbc.co.uk/rss/newsonline_uk_edition/
front_page/rss.xml".


The NASA button is in the Newsroom. Instead of
pushing the NASA button: say "Bang!"; request
"www.nasa.gov/rss/breaking_news.rss".


The WHO button is in the Newsroom. Instead of
pushing the WHO button: say "Bing!"; request "http://
www.doctorwhonews.net".
```

```
A screen is in the Newsroom.


Instead of examining the screen:
    if ready to read the file of RSS Responses, say "From
the screen you read:[line break][text of the file of RSS
Responses]";
    otherwise say "The screen remains blank for now."
```

As far as the story file is concerned, then, it sends a request down the communication line by writing the chosen RSS feed to the file named "rssrequest", and expects a reply to come back down the line by being written to the file "rssreply". However, the story file needs to expect that this might take some time. (Maybe forever, if there is no program responding, or if the Internet connection is not working.) The story file marks the "rssreply" file as not ready before it makes a request; if it subsequently finds that the file is now ready, that must mean that the other program has done the honours, and that all is well. In the mean time, "The screen remains blank for now."

Now for the RSS-SCRIPT program. The following provides a crude but workable program suitable for running as a Perl script on a system which provides the standard Internet fetching program "curl": Mac OS X, for instance. (If you have OS X, you can paste the following into a (Unix-format) text file called "rss-script.pl", place it in your home folder, open the Terminal utility, type "perl rss-script.pl", and then hide the Terminal window again.)

```
for (;;) { # repeat forever:
    system("sleep 1"); # wait 1 second
    open REQUEST, "rssrequest.glkdata" or next;
    # the request file has been detected:
    $header_line = <REQUEST>; # the header line
    $rss_feed = <REQUEST>; # the actual content - the
RSS feed URL
    close REQUEST;
    if ($header_line =~ m/^\*/) { # if the request file is
marked ready
        $rss = system("curl $rss_feed >rawrss.txt"); #
download the RSS feed
        # read the RSS XML into a single Perl string:
        open RAWRSS, "rawrss.txt" or next;
        $raw = "";
        while ($nl = <RAWRSS>) {
```

```
            $raw = $raw." ".$nl;
        }
        close RAWRSS;
        # look for the title and description of the first item:
        if ($raw =~ m/\<item\>\<title.*?\>(.*?)\<\/title\>.*?
\<description.*?\>(.*?)\<\/description\>/) {
            # write the reply:
            open REPLY, ">rssreply.glkdata" or next;
            print REPLY "* //RSS-SCRIPT// rssreply\n", $1,
"\n", $2, "\n";
            close REPLY;
            # request safely dealt with, so we can remove it:
            system("rm 'rssrequest.glkdata'");
        }
    }
}
```

Chapter 13: Testing and Publishing

§13.1. Testing; *§13.2. Publishing*

Contents of *The Inform Recipe Book*

Chapter 12: Typography, Layout, and Multimedia Effects

Indexes of the examples

§13.1. Testing

There are no recipes for testing, alas, although most experienced IF authors have their preferred ways of going about it. Briefly: the best advice is to build and maintain a Skein which holds complete play-throughs of the piece, so that it is easy to check with a single click that all is still well; to have beta-testers who play through (substantial) drafts and send back their transcripts of play, along with suggestions; and to listen to these suggestions, and treat the beta-testers as editors rather than galley slaves.

Alpha shows a way to gracefully accept beta-testers' annotations without advancing time in the story.

Most large works of IF have historically provided secret commands for testing and debugging - commands removed (or sometimes accidentally not) in the final released product. Inform does this automatically: the commands SHOWME, ACTIONS and SCENES are always present except in a released story file. It also allows us to write passages of source text which apply only for the testing phase, so that we can define new testing commands, or other checks that all is well: **Bic** demonstrates this, and is also useful in its own right.

Start of Chapter 13: Testing and Publishing

Back to Chapter 12: Typography, Layout, and Multimedia Effects: §12.5. Glulx Multimedia Effects

Onward to §13.2. Publishing

Example 2: ★ **Bic** Testing to make sure that all objects have been given descriptions.

Example 414: ★ **Alpha**
Creating a beta-testing command that matches any line starting with punctuation.

---

§13.2. Publishing

These three examples simply show what can be done using Inform's EPS-format map output, if one is willing to tweak the design in some vector-art program. Unfortunately, at present, there are few alternatives to Adobe Illustrator - a superb but very expensive program - in the field of EPS editing, and indeed, of vector art generally: this is especially the case for Windows users. Inkscape is a usable free alternative, but it needs to have EPS files translated to PDFs before they can be used. On Mac OS X, the built-in Preview application can do this; otherwise the open-source Ghostscript might be used, but it can be a pain to install. Still, for the IF author who does have EPS editing facilities available, Inform will play nicely with them.

Start of Chapter 13: Testing and Publishing

Back to §13.1. Testing

Example 446: ★ **Baedeker**
Creating a floorplan of the cathedral using the locations from previous examples.

Example 447: ★ **Port Royal 5**
Port Royal scenario given instructions for an EPS map.

Example 448: ★ **Bay Leaves and Honey Wine** Creating a map of Greece using the locations from previous examples.

Examples from Chapter 13: Testing and Publishing

Start of this chapter

Indexes of the examples

Testing to make sure that all objects have been given descriptions.

It may occasionally be useful to check whether all objects in our game have a given property. Here we have a "not for release" section that will run at the start of the game and alert us to any objects lacking description:

```
"Bic"

Section 1 - Testing descriptions - Not for release

When play begins (this is the run property checks at the
start of play rule):
    repeat with item running through things:
        if description of the item is "":
            say "[item] has no description."

Section 2 - Story

The Staff Break Room is a room.

The player carries an orange, a Bic pen, and a napkin.
The description of the orange is "It's a small hard pinch-
skinned thing from the lunch room, probably with lots of
pips and no juice."

The description of the napkin is "Slightly crumpled."
```

Sometimes we want to let testers of a game insert their own comments during a transcript, without those comments wasting turns of the game or producing lengthy or inappropriate parser errors. Many testers have a habit of prefacing comments with a punctuation mark, so let's say that we'd like to catch any command that starts with any punctuation at all:

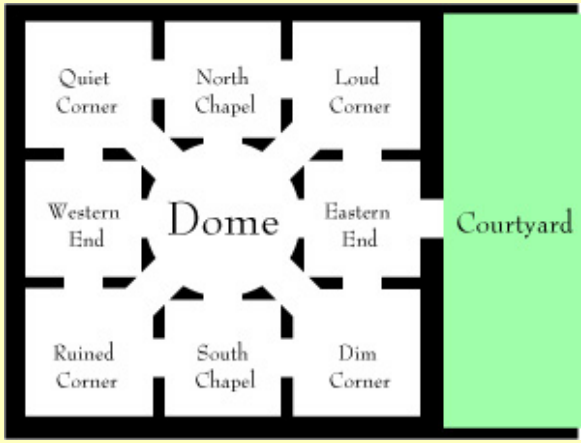
```
"Alpha"

When play begins:
    say "Hi, Larry! Thanks for testing my game!!"

Unimplemented Room is a room. "Room description
goes here..."

The scary troll is a man in Unimplemented Room.

After reading a command (this is the ignore beta-
comments rule):
    if the player's command matches the regular
expression "^\p":
        say "(Noted.)";
        reject the player's command.

Test me with "x me / x troll / !this game is a bit dull so far
/ kiss troll / ? does this troll do anything? / :yawn".
```

If our map is largely or entirely set inside a single building, we might want to produce something that resembles a floorplan. It's possible to do this with a little tweaking, like so:

```
"Baedeker"

Dome is a room. North of Dome is North Chapel. South
of the Dome is South Chapel. West of the Dome is
Western End. Quiet Corner is northwest of the Dome,
north of Western End, and west of North Chapel. Loud
Corner is east of North Chapel, northeast of Dome, and
north of Eastern End. Eastern End is north of Dim
Corner and east of Dome. Dim Corner is southeast of
Dome and east of South Chapel. Ruined Corner is
southwest of Dome, west of South Chapel, and south of
Western End.

The church door is east of Eastern End and west of the
Courtyard. The church door is a door.

Index map with
   room-shape set to "square" and
   room-size set to 60 and
   room-name-size set to 9 and
   room-name-length set to 13 and
   route-thickness set to 20 and
   room-outline set to off and
   map-outline set to off and
   route-colour set to "White" and
   room-colour set to "White" and
   room-shape of Dome set to "circle" and
   room-size of Dome set to 80 and
   EPS file.
```

Now we have a map made of white lines and boxes over a white background, which is not very exciting. If, however, we put a layer of

black under this and slightly adjust the room shapes (using an image editor such as Adobe Illustrator), we can produce something that plausibly resembles a floorplan or museum map, like so:



---

447 **ExamplePort Royal 5** *WI*

Port Royal scenario given instructions for an EPS map.

"1691"

Fort James is a room.

Thames Street End is south of Fort James.

Lime Street is south of Thames Street End. West of Thames Street End is north of Fisher's Row. The description of Fisher's Row is "A waterfront street that runs south towards Chocolata Hole, where the small craft are harboured. It also continues north around the tip of the peninsula from here, turning into the east-west

Thames Street."

Water Lane is east of Thames Street End.

East of Water Lane is a room called Thames Street at the Wherry Bridge. Thames Street at the Wherry Bridge has the description "To the southwest is the fishmarket; directly across the street is the entrance to a private alley through a brick archway."

The Fishmarket is southwest of Thames Street at the Wherry Bridge.

The Private Alley is south of Thames Street at the Wherry Bridge.

Thames Street by the King's House is east of Thames Street at the Wherry Bridge.

Thames Street before Fort Carlisle is east of Thames Street by the King's House.

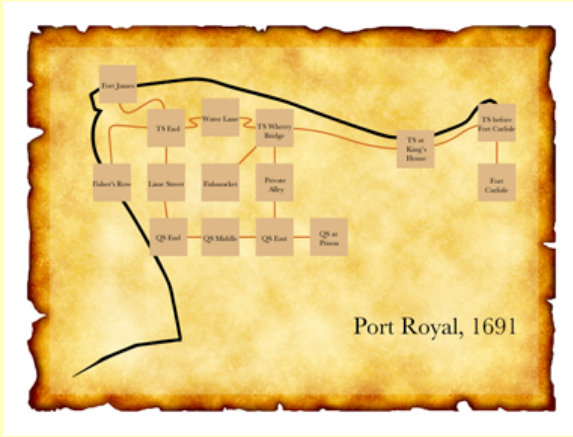
South of Thames Street before Fort Carlisle is a room called Fort Carlisle. The description of Fort Carlisle is "Handsomely arrayed with cannons which you could fire at any moment -- though of course there are ships at dock which might be in the way."

Queen Street End is south of Lime Street.

```
Queen Street Middle is east of Queen Street End.


Queen Street East is east of Queen Street Middle and
south of Private Alley.


Queen Street at the Prison is east of Queen Street East.


Index map with an EPS file and
    Fisher's Row mapped southwest of Thames Street
End,
    room-size set to 50 and room-name-size set to 10,
    room-name-length set to 15,
    route-thickness set to 2,
    room-outline set to off,
    map-outline set to off,
    route-colour set to "Chocolate",
    room-colour set to "Burly Wood",
    title set to "Port Royal, 1691",
    font set to "Baskerville",
    room-offset of Thames Street by the King's House set
to 160&-40,
    room-offset of Thames Street before Fort Carlisle set
to 210&10,
    room-offset of Fort Carlisle set to 210&0,
    room-offset of Fort James set to -90&-20,
    room-offset of Water Lane set to 0&20,
    room-offset of Queen Street End set to 5&0.
```

This sentence has become a long catalogue of specifications. To break it down: we disambiguate the placement of Thames Street End, which otherwise is hard to locate because the directions to and from the room are not symmetrical. Then we apply some general rules about size, font, and color. Finally, we add instructions about offsetting the room locations of a few specific rooms.

This last part is a bit finicky and will not be necessary in many cases, but our goal this time is to create a map diagram that can be superimposed on the real coastal outline of Port Royal at the time. With a bit of editing, the result looks like this:



Example **Bay Leaves and Honey Wine**

Creating a map of Greece using the locations from previous examples.

The map-maker can be used in quite versatile ways, in short; though the default is a schematic line-and-box affair, that is hardly the only option. While the EPS created is not always the result of our dreams, Inform usually can be made to do most of the hard and boring part, leaving the author to do only a bit of aesthetic touchup.

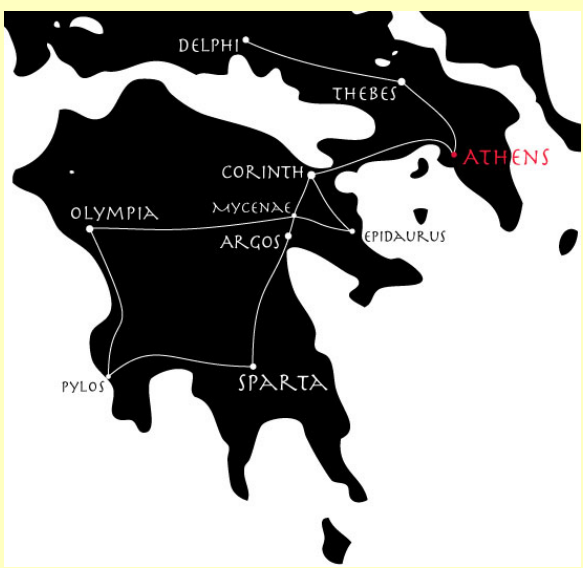
In many previous examples, we have sent hapless deities wandering around a map of Greece; we might like to chart that for ourselves, in a semi-realistic fashion. So:

```
"Bay Leaves and Honey Wine"

Corinth is a room. Athens is east of Corinth. Epidaurus
is southeast of Corinth and east of Mycenae. Mycenae
is south of Corinth. Olympia is west of Mycenae. Argos
is south of Mycenae. Thebes is northwest of Athens.
Pylos is south of Olympia. Sparta is east of Pylos and
south of Argos. Delphi is northwest of Thebes.
```

Index map with an EPS file and
room-size set to 8,
map-outline set to off,
room-name-offset set to 40&-40,
room-outline set to off,
room-colour set to "White",
route-colour set to "White",
room-name-colour set to "White",
room-name-length set to 25,
room-shape set to "circle".

This produces a line-and-dot map, where the names of rooms do not appear inside the city-circles, but rather (thanks to "room-name-offset") off to one side. We specify a long room-name-length because we want all the names of the cities spelled out in full; and we make all the elements white because we intend to place them over a black background layer.

We can then superimpose this on a vector map of Greece and tweak the exact positions of cities a little by hand (in Adobe Illustrator, as it happens, but other programs would also allow this level of editing). The result:



Alphabetical Index of Examples

Examples in Thematic Order
Examples in Numerical Order
General Index
Part I. Writing with Inform
Part II. The Inform Recipe Book

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A
A&E
AARP-Gnosis
The Abolition of Love
About Inform's regular expression support
About the examples
Accented and exotic letters and symbols
Access All Areas
Actaeon
Actions prohibited in the presence of a dangerous character
Actor's Studio
Adverbs used in commands
Aftershock
Age of Steam
Ahem
Air Conditioning is Standard
Alias
Alien Invasion Part 23
All actions given different durations
All Roads Lead to Mars
Allowing EXAMINE to see multiple objects with a single command
Allowing the player to describe his character before starting play
Alpaca Farm
Alpha
Altering response text programmatically
America Stands Tall
AMUSING menu shown at the endgame
Anchorite
Annoyotron Jr
Another, distinct from the player
Ant-Sensitive Sunglasses
Anteaters
Apples
Appointments at future times

Appraisal
The Art of Noise
ASK person TO do something, understood
ASK someone ABOUT an object the same as SHOW
ASKing about a conversation topic other than the one the player typed
Asking the player a Yes/No question
Asking the player to choose a gender
Aspect
Assigning attributes randomly at the start of play
ATTACK and DROP break fragile things
Attitude rules
An automatically closing door
Awarding points for visiting a room

Backdrops that move conditionally
Backdrops which can only be examined
Backus-Naur form for rules
Bad Hair Day
Baedeker
Baking soda which has a different name when next to a smelly thing
Ballpark
Banner printing at appropriate times
Baritone, Bass
Barter Barter
Basic room, container, and supporter descriptions
Basic switchable light for a room
Basic window similar to a door
Battle of Ridgefield
Bay Leaves and Honey Wine
Beachfront
Bee Chambers
Beekeeper's Apprentice
Being Peter
Being Prepared
Belfry
Beneath the Surface
The Best Till Last
Beverage Service
Bibliophilia
Bic
The Big Sainsbury's
Big Sky Country
Bikini Atoll
Blackout
Blanking the status line before play
Blankness
Blink
Bogart

Book with numbered pages
Book with various contents
Bookshelf with numerous books
Bosch
Boston Cream
Bottles with removable stoppers
Bowler Hats and Baby Geese
Bribery
Broken and unbroken flowerpots
Brown
Bruneseau's Journey
Bumping into Walls
BURN command, and flammable objects
Buttons
BUY command and a system of money
BUY command and money, complete with denominations

C
Cactus Will Outlive Us All
Camp Bethel
Campfire lit
Candy
Capital City
Capitalised status line
Car which can only be driven on roads
Carnivale
Carousel Room from which exits lead to random locations
Case change for text produced by to say phrases
Casino Banale
Cave Entrance
Cave-troll
Celadon
Centered
Chanel Version 1
Change of Basis
Changing the final question after victory
Channel
1. Television that can be referred to by channel
2. Television with more advanced parsing
Character who learns new actions by watching the player
Characters and objects with titles and special articles
Characters killed off by trigger actions
Characters referred to when absent
Characters renamed in play
Check rules modified for taking something we're inside
Checking for missed properties
Cheese-makers
Chest with a supporting lid
Child who asks if we're there yet

Chronic Hinting Syndrome
Cinco
Circle of Misery
Claims Adjustment
Cloak of Darkness
Clocks that can be SET TO any time
Clothes made of priced fabrics
Clothing kinds
Clothing that layers
Clothing with layering and regions of the body
Clouds with random descriptions
Cloves
Clueless
Colour names for exotic pigments
COMBINE action for connecting multiple objects into one
Command applying to an object added to the story
Command prompt changing to reflect state of combat
Commercial break
Compass directions renamed
Complimentary Peanuts
Concealed pet who would yip at you if it could see you
Concealing clothing for other characters
Contact with objects detected
Containers and supporters given a common kind
Containers referred to by contents
Containers that calculate internal volume and available room
Containers which have a breaking strain
Continuous space with distant objects visible
Continuous space with distant objects visible and automatic room description
Continuous space, simple
Control Center
Controlling the order in which multiple-object actions occur
Conversation based on keyword recognition
Conversation system with recap of past exchanges
Conversation where characters seek logical connection to foregoing topics
Conversation with multiple options for each conversation topic
Copper River
Costa Rican Ornithology
The Cow Exonerated
The Crane's Leg
1. Descriptions based on how objects compare with player's expectations
2. Descriptions based on player-character's observation abilities

Creating our own text variations rules
Creating our own text variations rules, part two
Croft
Crusoe
Curare
CUT and TIE for lengths of string

CUT [something] WITH [something] command
Cycle of day and night scenes
Cyclical randomization of named objects

D
Dangerous objects not to be touched
The Dark Ages Revisited
Dark rooms given non-standard description
A Day For Fresh Sushi
Day One
Deadbolted door unlockable without a key on one side
Dearth and the Maiden
Death message replaced
Deck of cards
Delayed Gratification
Delicious, Delicious Rocks
Democratic Process
Depth
Describing action outcomes using a random choice of verb
Describing action outcomes using a random choice of verb or describing the world state that results
Describing objects with parts
Describing ongoing character behavior with participles
Describing relations using a random choice of verb
Description from inside a vehicle
Device to remember and play back actions
DIAGNOSE command
Dimensions
Dinner is Served
Disambiguating based on properties
Disambiguating body parts
Disambiguation question giving items more explicit names
Disambiguation question worded in a new way entirely
Disenchantment Bay
1. The charter boat
2. Scenery
3. Glacier view
4. Descriptions of things
5. Cabin door and deck
6. Locked-up fishing rods
7. Radar and instruments
8. Pushable chest of ice
9. The Captain
10. Clothing and possessions
11. The player's backpack
12. Complete playable scenario

Disorderly conduct
Do Pass Go
Don Pedro's Revenge

Door described and parsed differently depending on where it is
Door described differently depending on where it is
Door kind that describes the destination
Doors as seen by NPCs
Doors that open automatically
Down Below
Down in Oodville
Drawers where thing is always in the last opened
DRINKing a potion which then goes away
DROP applies even to objects the player carries indirectly
DROP modified to distinguish between large and small enterable supporters
DROPPING into and onto things
Dubai
Duplicated blue, red and green building blocks

Early Childhood
Eddystone
Electric light kind of device
Electrified
An elevator operated simply by walking in and out
An elevator which takes the player to alternate floors
Elsie
Emma
Empire
Encyclopedia set
Endurance
Enterable objects
Entrapment
Entrevaux
Entropy
Equation used for price curves
Equipment List
Escape
Escape action for non-player characters only
Escape from the Seraglio
Every turn an opponent plays
Exact numbers replaced with vague quantifiers
EXAMINE multiple things at once
EXAMINE produces special results for a group of objects
Examining everything at once
Examining the player
An example bouquet of flowers
Exits added to a room
Exits and rooms listed in the status line
Exits only listed in the status line
Extension design advice
Extra Supplies
The Eye of the Idol

Eyes, Fingers, Toes

Fabrication
The Facts Were These
Falling into a pit when going from darkness to darkness
Fallout Enclosure
Farewell
Fate Steps In
Faux-infinite supply of red pens
Feline Behavior
Ferragamo Again
The Fibonacci Sequence
Fido
The Fifth Body
Fifty Times Fifty Ways
Fifty Ways to Leave Your Larva
Filtering text output in room names
Finality
FIND command
Finding the bottom of a pile
Fine Laid
Finishing School
First look text added for a newly entered room
First Name Basis
Fish of ambiguous species
Flashlight
Flathead News Network
Flotation
FOLLOW command
Food Network Interactive
Food with ingredients affecting the player
Footnote system
For Demonstration Purposes
Fore
Formal syntax of sentences
Formicidae
Four Cheeses
Four Stars

1. Dark room which relies on other senses when light is off
2. Scope for listening different from scope for seeing

The Fourth Body
Fragment of a Greek Tragedy
Frizz
Frozen Assets
Full Moon
FULL SCORE using a list of stored actions
FULL SCORE with achievements table and rankings
Fun with Participles

Furniture, resentment of
Further Reasons Why All Poets Are Liars

G
aribaldi
1. Readout of all doors in the game
2. Coloured text styles

Generation X
Get Axe
GET DOWN and DOWN understood as EXIT
Get Me to the Church on Time
Ginger Beer
GIVE action for other characters
GIVE command extended
GIVE extended to multiple objects at once
GO BACK command
GO equivalent to GO OUT
GO OUT and GO IN determine most appropriate direction if none is defined
GO TO ROOM command
GO TO ROOM with automatic door-handling en route
GO UP and GO DOWN determined by room altitude
Goat-Cheese and Sage Chicken
GOing an invalid direction produces a list of valid exits
Going Going
Gopher-wood
The Gorge at George
Graph-theory view of relations
Grilling
Guided Tour

H
he Hang of Thursdays
Happy Hour
Hatless
A Haughty Spirit
Hays Code
Hayseed
Heat death
HELP command is recommended if the player seems lost
HELP command with specific set of topics
HELP menu from extension, with added content
HELP with a simple menu
A hexagonal map grid for movement
Hiding things under other things
Higher Calling
Hint access able to be turned off for the duration of the game
Hints leading the player through conversation
Hints leading the player through physical puzzles
Histories of previous games saved to external file
History Lab

Hohmann Transfer
Horribly heavy box described differently after being taken
Hot Glass Looks Like Cold Glass
Hotel Stechelberg
Hour of the Wren
Hover
HTML-style italic and boldface tags
Hudsucker Industries
A Humble Wayside Flower
Hunger eventually killing the player
Hymenaeus
The Hypnotist of Blois

I Didn't Come All The Way From Great Portland Street
Ibid.
Identifying nouns by the last thing done to them
Identity Theft
If It Hadn't Been For...
Igpay Atinlay
Implicit takes require time
In Fire or in Flood
Incorporating changeable objects into the body of the room description
Indirection
Infiltration
Instant EXAMINE and LOOK
Interrogation
Introduction to Juggling
INVENTORY handling of properties made more flexible
INVENTORY listing with separate carried and worn items
INVENTORY revised for other characters
Inventory standard response changed
INVENTORY wide and tall
IPA
IQ Test
Ish.

Jamaica 1688
Joke command added
Jokers Wild

Keywords in place of ordinary commands
Kinds given new singular and plural synonyms
Kiwi
Kyoto

Labyrinth of Ghosts
Lakeside Living

Lanista
    1. Simple Randomized Combat
    2. Randomized Combat with Weapons
Latin Lessons
Latris Theon
Laura
Lead Cuts Paper
Lean and Hungry
The Left Hand of Autumn
Lemonade
Leopard-skin
Lethal Concentration
    1. Gas diffusion from a source through adjacent rooms
    2. Gas diffusion in which gas is heavier than air
Letters described as a group
Levels of lighting
Lies
Light levels variable depending on the number of candles the player has lit
Light-filled rooms as a route through a space
Lighting and relighting a candle
Liquid containers with measured contents
Liquid model with large bodies of liquid added
Liquids in mixtures, identified by recipe
LISTEN command extended
LOCK and UNLOCK with altered carrying requirements
Lollipop Guild
LOOK [direction] command
LOOK mentions only currently useful objects
LOOK UNDER shows nothing unless the player has a light
Low Light
Lucy
Lugubrious Pete's Delicatessen

M. Melmoth's Duel
Machine that transmutes objects
Magnet which picks up nearby metal objects
Magneto's Revenge
The Man of Steel
The Man of Steel Excuses Himself
Map region listed in status line
Marble chute toy
Masochism Deli
Matches that set fires
Mathematical view of relations
Matreshka
Mattress King
Maze escaped by performing an exact sequence of actions
Maze with randomized room links

Meet Market
Menu of numerical options
Merlin
Meteoric I and II
Midsummer Day
Mimicry
Ming Vase
Minimal Movement
Mirror into which the player can look
Mirror remembering a room description from the past
Mirror, Mirror
Misadventure
Missing nouns supplied in commands to characters
Mistress of Animals
Model objects referred to by the thing modeled
Model Shop
Modern Conveniences
Modifying and re-parsing an entered command
Money for Nothing
Morning After
Moving a backdrop during play
Mr. Burns' Repast
MRE
Multiple-object actions given modified descriptions
Murder on the Orient Express
Murderer chosen randomly at start of play
Mutt's Adventure

N

Name of a dog set by the player
Name of player character selected at start of play
Nameless
Names of objects change with player's diction
Narration which changes as the player learns new facts
Narrative Register
Near-infinite supply of pizza
Neighborhood Watch
Nickel and Dimed
Night and Day
The Night Before
Night Sky
Nine AM Appointment
No Place Like Home
No Relation
Noisemaking
Noisy Cricket
North by Northwest
North-northwest and similar directions
Northstar

Novice mode offers suggestions before each prompt
Number Study
Numberless
Numbers in relations

Object-based rulebooks
Observation stage after report
Odin
Odyssey
OFFER price FOR command allowing player to bargain
Olfactory Settings
On/off button for devices
One of Those Mornings
One Short Plank
Only You...
Orange Cones
Originals
Otherwise if demonstrated
Otranto
Owen's Law
Oyster Wide Shut

Paddington
Pages
Pair of dice for rolling
Palette
Panache
Paragraphs of flexible descriptions
Parser errors involving the any token
Past tense conditions
Patient Zero
Peeled
Peers
People changing their titles during play
People commanded to obey
People introduced by relative
People listed as a group, with their possessions afterward
People perform errands and randomly spread disease
People who do new things each time the player looks
People who interact with each other each turn
People who move around a party on their own
People who must be greeted before conversation
People who reject categories of instruction
Persephone
Person capable of reaching through solid objects
Person who answers questions based on a common pool of knowledge
Person who answers what, where, and when questions differently
Person who comments on the player's every action

Person who finds a path to a destination specified by player
Person who follows the player
Person who gets fed up after being asked to do many implausible things
Person who laments broken objects
Person who moves on a pre-determined path
Person who moves randomly
Person who obeys almost all instructions
Person who opens a container before trying to get something from it
Person who picks up treasures
Person who pursues his own goals each turn
Peugeot
Photographs referred to by subject matter
Phrase to repeat a rule
Phrases applying to specific things
Pig Latin for the player's commands
Pine

1. Waking the Princess
2. Chatting with the Princess
3. Flashback scenes described to the Princess
4. Flashback with multiple endings

Pink or Blue
Pizza Prince
Plank bridge breaks on being crossed when the player is carrying something
Player controls multiple player-characters in turn
Player's erroneous command recorded for later
Player's height randomized at the start of play
Plot summary in place of a score
Pocket added to every jacket
A point for never saving the game
The Pointy Hat of Liminal Transgression
Polarity
Police State
Politics as Usual
Polygons
Port Royal

1. A landscape from Jamaica, 1691
2. With one-way connections added
3. Divided into regions
4. Character names
5. Producing an EPS map

Postures for sitting, standing, and lying down
Pot of Petunias
POUR that pours liquid on a fire
The Power of the Keys
The Prague Job
Preferences file loaded on replaying
Princess and the Pea
Printed names for complex-named objects

Printing the description of something activity
Priority Lab
Prisoner's Dilemma
The Problem of Edith
Producing an EPS floorplan
Producing an EPS map of Greece
Prolegomena
Properties, checking for
Proposal
Provenance Unknown
Puff of Orange Smoke
Puncak Jaya
PUSH something A DIRECTION modified
PUSH something a direction with extra phrasing
PUSH SOMETHING TO check rules modified
Pushing a box between named internal positions in a room
PUT and INSERT automatically TAKE first
PUT and INSERT automatically TAKE first even with multiples

Q
The Queen of Sheba
Query
Questionable Revolutions
Questions asked of the player
Quiz Show

R
Radio Daze
Radio producing cycling programming
Radios and other devices active when switched on
Raised supporter kind that conceals its contents
Random distribution at the start of play
Random variations within text
Randomized pedestrian passer-by
rBGH
READ command separate from EXAMINE
Real Adventurers Need No Help
Recorder that records sounds made by player and non-player actions
Red Cross
Red sticky label
Redirecting actions to new objects
Redirecting all actions from one object to another
Reflections
Refusing to understand "hot"
Region off-limits to a player without VIP pass
Regions, traveling from
Relations and their uses
Relations applying to multiple values of the same kind
Relations syntax explored
Relevant Relations

REMEMBER command keyed to topics
Removal
Reordering multiple objects for dramatic effect
Replanting
Reporting rules for other characters' behavior
Responding to questions starting with WHO, WHAT, etc.
Responsive
Restrictions preventing inappropriate behavior
Results of a rule
Resuming play after an accidental death
Revenge of the Fussy Table
Rip Van Winkle
Robo
1. Robot that stores only one program at a time
2. Robot able to store and perform fifteen named scripts

Rock Garden
Rocket Man
Room description changes after first visit
Room description changes after note is read
Room description changes at each of several visits
A room where the game cannot be saved
Rooms player is forced to visit in order
Rooms with sounds of nearby water, illuminated by flickering light
Rooms with sounds of nearby water, illuminated by flickering light
Rooms with visible exteriors
Rope, able to be tied to things and dragged betwixt rooms
An RSS reader connecting to the Internet
Rubies
Rules of Attraction

S
Sackcloth
Safari Guide
Safe that can be opened with a combination
Safe with a multi-number combination
Safety
Saint Eligius
Sand
Sanity-check stage before the Before rules
Savannah
Saying a number in round numbers
Scene that ends as soon as the player has embarrassed himself
Scene which plays through a series of events in order, then ends
Scene which starts and finishes
Scene with random atmospheric events
Scenes which change as the player explores his inventory
Scenes which restrict movement
Scenes with props lists and location properties
Scheduling an eclipse

Scope approaches compared
Scope for large objects visible far away
Scoreboard preserved between games
Scored items listed in a table
Search and Seizure
SEARCH [room] action that opens every visible unlocked container
SEARCHing for an item which moves into play
The Second Oldest Problem
Setting pronouns to respond to sudden changes
SHAKE command
Shawn's Bad Day
Shifting alliances among people
Shiny items are described as reflecting torchlight
Shipboard directions
Shipping Trunk
Shirts constructed from component parts
SHOOT command, and ray gun disintegrating objects
Shops open and close at specific hours
SHOW something without taking it first
Shuffling the arrangement of items in a container
Sieve of Eratosthenes
Signpost that can be turned
Signposts to distant rooms
Signs and Portents
Single yellow onion in a sack
Situation Room
Sleep and waking
Slightly Wrong
Slogar's Revenge
Slouching
Smart window
Smoke which spreads, not tracking concentration
Smuggler carrying hidden items
Snip
Solitude
Some Assembly Required
Something Narsty
Space Patrol - Stranded on Jupiter!
The Speed of Thought
Spellbreaker
Spring Cleaning
Staircase kind of door which diverts the CLIMB command
Staircase kind of door, always open and unopenable
Standard IF demonstration
Starry Void
Stately Gardens
Status line with centered text, the easy way
Status line with centered text, the hard way

Stone
Story driven by internal rules
Straw Boater
Straw Into Gold
Streaming
Strictly Ballroom
Studious
Substituting insults and boasts
Supporter from which the player cannot take things
Supporters that don't list contents in room descriptions
Sweeney
Swerve left? Swerve right? Or think about it and die?
Swigmore U.
Sybil

1. ASK, TELL, and ANSWER conflated
2. YES and NO directed to people

Table of juggling equipment
TAKE action reports where the noun came from
TAKE prints a description
TAKE scenery producing a custom response
TALK TO command used with scenes
TALK TO usage corrected
Tamed
Technological Terror
Telephones reaching distant parties
Telephones with standard American-length phone numbers
Telescope allowing view of another room
Tense Boxing
Terracotta
Terracottissima
Terracottissima Maxima
Terror of the Sierra Madre
Testing command
Testing for empty properties
Testing the locale description order
Text Foosball
Thirst
Thirst 2
3 AM
THROW [something] AT [something] command enhanced
Tilt

1. Deck of cards that can be referred to by name
2. Identifying poker hands
3. Playing cards with colored unicode suits

Time expressed in units other than minutes
Time told in 24-hour military style
Timeless

Tiny Garden
To say rules for insults and boasts
Today Tomorrow
Tom's Midnight Garden
Torch understood as flaming or extinguished
Totality
Tour des Maillots
Trachypachidae Maturin 1803
Train that makes station stops
Transmutations
Travel requiring a vehicle
Travel to a room described
Trieste
The Trouble with Printing
Trying Taking Manhattan
Turns take 15 minutes each
Turns take a quarter day each

U-Stor-It
Unblinking
The Unbuttoned Elevator Affair
Uncommon Ground
Under Contract
Understanding names of things, basic
Understanding new commands involving compass directions
Understanding text
Understanding units as part of the name of a thing
The Undertomb
The Undertomb 1
The Undertomb 2
UNDO not mentioned in the final question
The Unexamined Life
Unit conversions
Unpeeled
Unthinkable Alliances
Up and Up
Uptempo
Uptown Girls
USE action which divines rational behavior for a wide range of possible nouns
Using lists as arrays
Using lists as queues
Using lists as rings
Using lists as sets
Using lists as sieves
Using lists as stacks

Van Helsing
Variety

Variety 2
Variety of postures
Varying room description text
Verbosity
1. Brief descriptions by default
2. Full-length descriptions mandatory

Veronica
Versailles
Video camera that records all actions
A View of Green Hills
Virtue
Vitrine
Vouvray

W
Wainwright Acts
WAIT [number] MINUTES command
WAIT UNTIL [time] command
Waiting Room
Walls and Noses
Wand which reveals a person's concealed possessions
Waterskin being depleted
Waterworld
Ways Out
Weathering
Weighbridge
Wesponses
Western Art History 305
What Makes You Tick
What Not To Wear
When?
Whence?
Whither?
Widget Enterprises
Window connecting two rooms
Windows in high places
Winds of Change
Witnessed
1. Batteries, swappable between devices
2. EMF Meter

Wonderland
Wonka's Revenge
The World of Charles S. Roberts
Would you...?
WRITE IN command that will open a free-text mode
WRITE text IN command that will store text
WXPQ

Xerxes

Xot
Xylan
XYZZY

Y ask Y?
Yolk of Gold
You see nothing special... message replaced
Your Mother Doesn't Work Here

Zero
Zodiac
Zorb
Zork II
Zorn of Zorna
Zqlran Era 8

The Über-complète clavier

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Examples in Numerical Order

Examples in Alphabetical Order
Examples in Thematic Order
General Index
Part I. Writing with Inform
Part II. The Inform Recipe Book

Chapter 1: Welcome to Inform

§1.1. Preface

1 **Example** **About the examples** *RB*
An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

Chapter 3: Things

§3.1. Descriptions

2 **Example** **Bic** *RB*
Testing to make sure that all objects have been given descriptions.

3 **Example** **Verbosity 1** *RB*
Making rooms give brief room descriptions when revisited.

4 **Example** **Slightly Wrong** *RB*
A room whose description changes slightly after our first visit there.

§3.2. Rooms and the map

5 **Example** **Port Royal 1** *RB*
A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

6 **Example** **Up and Up** *RB*
Adding a short message as the player approaches a room, before the room description itself appears.

7 **Example** **Starry Void** *RB*

Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

§3.3. One-way connections

8 **Example** **Port Royal 2** *RB*

Another part of Port Royal, with less typical map connections.

9 **Example** **The Unbuttoned Elevator Affair** *RB*

A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

§3.4. Regions and the index map

10 **Example** **Port Royal 3** *RB*

Division of Port Royal into regions.

§3.5. Kinds

11 **Example** **First Name Basis** *RB*

Allowing the player to use different synonyms to refer to something.

12 **Example** **Midsummer Day** *RB*

A few sentences laying out a garden together with some things which might be found in it.

§3.6. Either/or properties

13 **Example** **Tamed** *RB*

Examples of a container and a supporter that can be entered, as well as nested rooms.

§3.7. Properties depend on kind

14 **Example** **Disenchantment Bay 1** *RB*

A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

§3.8. Scenery

15 **Example** **Disenchantment Bay 2** *RB*

Disenchantment Bay: creating some of the objects in the cabin's description.

Example **Replanting**

Changing the response when the player tries to take something that is scenery.
16 *RB*

§3.9. Backdrops

17 **Example** **Disenchantment Bay 3** *RB*
Disenchantment Bay: adding a view of the glacier.

§3.11. Two descriptions of things

18 **Example** **Disenchantment Bay 4** *RB*
Disenchantment Bay: fleshing out the descriptions of things on the boat.

19 **Example** **Laura** *RB*
Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

§3.12. Doors

20 **Example** **Disenchantment Bay 5** *RB*
Disenchantment Bay: adding the door and the deck to our charter boat.

21 **Example** **Escape** *RB*
Window that can be climbed through or looked through.

22 **Example** **Garibaldi 1** *RB*
Providing a security readout device by which the player can check on the status of all doors in the game.

§3.13. Locks and keys

23 **Example** **Disenchantment Bay 6** *RB*
Disenchantment Bay: locking up the charter boat's fishing rods.

24 **Example** **Neighborhood Watch** *RB*
A locked door that can be locked or unlocked without a key from one side, but not from the other.

§3.14. Devices and descriptions

25 **Example** **Disenchantment Bay 7** *RB*
Disenchantment Bay: making the radar and instruments switch on and off.

26 **Example** **Down Below** *RB*
A light switch which makes the room it is in dark or light.

§3.16. Vehicles and pushable things

27 **Example** **Peugeot** *RB*
A journey from one room to another that requires the player to be on a vehicle.

28 **Example** **Disenchantment Bay 8** *RB*
Disenchantment Bay: a pushable chest of ice for the boat.

29 **Example** **Hover** *RB*
Letting the player see a modified room description when he's viewing the place from inside a vehicle.

§3.17. Men, women and animals

30 **Example** **Disenchantment Bay 9** *RB*
Disenchantment Bay: enter the charter boat's Captain.

§3.18. Articles and proper names

31 **Example** **Belfry** *RB*
You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

32 **Example** **Gopher-wood** *RB*
Changing the name of a character in the middle of play, removing the article.

§3.20. Possessions and clothing

33 **Example** **Disenchantment Bay 10** *RB*
Disenchantment Bay: things for the player and the characters to wear and carry.

§3.21. The player's holdall

34 **Example** **Disenchantment Bay 11** *RB*
Disenchantment Bay: making a holdall of the backpack.

§3.23. Parts of things

Example **Fallout Enclosure**
Adding an enclosure kind that includes both containers and supporters in order to simplify

text that would apply to both.
35 *RB*

Example **Brown**
36 A red sticky label which can be attached to anything in the game, or removed again. *RB*

Example **Disenchantment Bay 12**
37 A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained. *RB*

§3.24. Concealment

Example **Search and Seizure**
38 A smuggler who has items, some of which are hidden. *RB*

§3.25. The location of something

Example **Van Helsing**
39 A character who approaches the player, then follows him from room to room. *RB*

§3.26. Directions

Example **Prisoner's Dilemma**
40 A button that causes a previously non-existent exit to come into being. *RB*

Example **The World of Charles S. Roberts**
41 Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape. *RB*

Example **Fore**
42 Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel. *RB*

Chapter 4: Kinds

§4.1. New kinds

Example **Vouvray**
43 Adding synonyms to an entire kind of thing. *RB*

§4.3. Degrees of certainty

44 **Example** **Odin** *RB*
Replacing "You see nothing special..." with a different default message for looking at something nondescript.

45 **Example** **Something Narsty** *RB*
A staircase always open and never openable.

§4.4. Plural assertions

46 **Example** **Get Me to the Church on Time** *RB*
Using kinds of clothing to prevent the player from wearing several pairs of trousers at the same time.

§4.7. New either/or properties

47 **Example** **Change of Basis** *RB*
Implementing sleeping and wakeful states.

§4.8. New value properties

48 **Example** **Would you...?** *RB*
Adding new properties to objects, and checking for their presence.

49 **Example** **Straw Boater** *RB*
Using text properties that apply only to some things and are not defined for others.

§4.9. Using new kinds of value in properties

50 **Example** **The Undertomb 1** *RB*
A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

51 **Example** **The Undertomb 2** *RB*
Flickering lantern-light effects added to the Undertomb.

52 **Example** **The Crane's Leg 1** *RB*
A description text that automatically highlights the ways in which the object differs from a standard member of its kind.

53 **Example** **Signs and Portents** *RB*
Signpost that points to various destinations, depending on how the player has turned it.

§4.12. Values that vary

54 ★★★ **Example** **Real Adventurers Need No Help** *RB*
Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

§4.14. Duplicates

55 ★★★ **Example** **Early Childhood** *RB*
A child's set of building blocks, which come in three different colours - red, green and blue - but which can be repainted during play.

§4.15. Assemblies and body parts

56 ★ **Example** **Being Prepared** *RB*
A kind for jackets, which always includes a container called a pocket.

57 ★★ **Example** **Model Shop** *RB*
An "on/off button" which controls whatever device it is part of.

58 ★★★ **Example** **U-Stor-It** *RB*
A "chest" kind which consists of a container which has a lid as a supporter.

59 ★★★ **Example** **The Night Before** *RB*
Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

Chapter 5: Text

§5.4. Text with numbers

60 ★★★ **Example** **Ballpark** *RB*
A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

§5.5. Text with lists

61 ★ **Example** **Control Center** *RB*
Objects which automatically include a description of their component parts whenever they are examined.

62 ★★ **Example** **Tiny Garden** *RB*
A lawn made up of several rooms, with part of the description written automatically.

§5.6. Text with variations

63 ★ **Example When?** RB
A door whose description says "...leads east" in one place and "...leads west" in the other.

64 ★★★ **Example Whence?** RB
A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

65 ★★★ **Example Persephone** RB
Separate the player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

§5.7. Text with random alternatives

66 ★ **Example Radio Daze** RB
A radio that produces a cycle of output using varying text.

67 ★★ **Example Camp Bethel** RB
Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

§5.8. Line breaks and paragraph breaks

68 ★★ **Example Beekeeper's Apprentice** RB
Making the SEARCH command examine all the scenery in the current location.

§5.9. Text with type styles

69 ★ **Example Garibaldi 2** RB
Adding coloured text to the example of door-status readouts.

§5.11. Unicode characters

70 ★★★ **Example The Über-complète clavier** RB
This example provides a fairly stringent test of exotic lettering.

§5.13. Making new substitutions

71 ★ **Example Fifty Ways to Leave Your Larva** RB
Using text substitution to make characters reply differently under the same circumstances.

72 ★★★ **Example Fifty Times Fifty Ways** RB
Writing your own rules for how to carry out substitutions.

Chapter 6: Descriptions

§6.4. Defining new adjectives

73 ★ **Example** **Finishing School** *RB*
The "another" adjective for rules such as "in the presence of another person".

§6.5. Defining adjectives for values

74 ★★★ **Example** **Only You...** *RB*
Smoke which spreads through the rooms of the map, but only every other turn.

§6.9. Which and who

75 ★★ **Example** **Versailles** *RB*
A mirror which will reflect some random object in the room.

§6.13. To be able to see and touch

76 ★★★ **Example** **Lean and Hungry** *RB*
A thief who will identify and take any valuable thing lying around that he is able to touch.

§6.14. Adjacent rooms and routes through the map

77 ★ **Example** **Mistress of Animals** *RB*
A person who moves randomly between rooms of the map.

78 ★ **Example** **All Roads Lead to Mars** *RB*
Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

79 ★★ **Example** **Hotel Stechelberg** *RB*
Signposts such as those provided on hiking paths in the Swiss Alps, which show the correct direction and hiking time to all other locations.

80 ★★★ **Example** **A View of Green Hills** *RB*
A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

81 ★★★ **Example** **Unblinking** *RB*
Finding a best route through light-filled rooms only, leaving aside any that might be dark.

§6.15. All, each and every

82 ★★★★ **Example** **Revenge of the Fussy Table** *RB*
A small game about resentful furniture and inconvenient objects.

§6.16. Counting while comparing

83 ★★ **Example** **Yolk of Gold** *RB*
Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

Chapter 7: Basic Actions

§7.2. Instead rules

84 ★ **Example** **Grilling** *RB*
A grill, from which the player is not allowed to take anything lest he burn himself.

85 ★ **Example** **Bad Hair Day** *RB*
Change the player's appearance in response to EXAMINE ME.

§7.3. Before rules

86 ★★★ **Example** **Democratic Process** *RB*
Make PUT and INSERT commands automatically take objects if the player is not holding them.

87 ★★★★ **Example** **Sand** *RB*
Extend PUT and INSERT handling to cases where multiple objects are intended at once.

§7.4. Try and try silently

88 ★ **Example** **Fine Laid** *RB*
Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

89 ★ **Example** **Hayseed** *RB*
A refinement of our staircase kind which can be climbed.

§7.5. After rules

90 ★ **Example** **Morning After** *RB*
When the player picks something up which he hasn't already examined, the object is

described.

§7.6. Reading and talking

91 **Example** **Sybil 1** *RB*
Direct all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

92 **Example** **Lucy** *RB*
Redirecting a question about one topic to ask about another.

93 **Example** **Sybil 2** *RB*
Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

94 **Example** **Costa Rican Ornithology** *RB*
A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

§7.7. The other four senses

95 **Example** **The Art of Noise** *RB*
Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

§7.9. All actions and exceptional actions

96 **Example** **Zodiac** *RB*
Several variations on "doing something other than...", demonstrating different degrees of restriction.

§7.10. The noun and the second noun

97 **Example** **Ming Vase** *RB*
ATTACK or DROP break and remove fragile items from play.

§7.12. In the presence of, and when

98 **Example** **Beachfront** *RB*
An item that the player can't interact with until he has found it by searching the scenery.

99 **Example** **Today Tomorrow** *RB*
A few notes on "In the presence of" and how it interacts with concealed objects.

§7.13. Going from, going to

100 **Example** **Veronica** *RB*
An effect that occurs only when the player leaves a region entirely.

101 **Example** **A&E** *RB*
Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

102 **Example** **Bumping into Walls** *RB*
Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

103 **Example** **Polarity** *RB*
A "go back" command that keeps track of the direction from which the player came, and sends him back.

§7.14. Going by, going through, going with

104 **Example** **No Relation** *RB*
A car which must be turned on before it can be driven, and can only go to roads.

105 **Example** **Mattress King** *RB*
Adding extra phrasing to the action to PUSH something in a direction.

106 **Example** **One Short Plank** *RB*
A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

107 **Example** **Provenance Unknown** *RB*
Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

108 **Example** **Zorb** *RB*
Replacing the message the player receives when attempting to push something that isn't pushable, and also to remove the restriction that objects cannot be pushed up or down.

§7.15. Kinds of action

109 **Example** **Dearth and the Maiden** *RB*
Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and seemliness.

Example **Mimicry**

People who must be greeted before conversation can begin.
110 *RB*

§7.16. Repeated actions

111 Example **Y ask Y?** *RB*
Noticing when the player seems to be at a loss, and recommending the use of hints.

112 Example **A Day For Fresh Sushi** *RB*
A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is performed.

Chapter 8: Change

§8.2. Changing the command prompt

113 Example **Don Pedro's Revenge** *RB*
Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

§8.3. Changing the status line

114 Example **Politics as Usual** *RB*
Have the status line indicate the current region of the map.

115 Example **Centered** *RB*
Replacing the two-part status line with one that centers only the room name at the top of the screen.

§8.4. Change of either/or properties

116 Example **Vitrine** *RB*
An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

§8.5. Change of properties with values

117 Example **Thirst** *RB*
A waterskin that is depleted as the player drinks from it.

118 Example **Thirst 2** *RB*
A campfire added to the camp site, which can be lit using tinder.

§8.7. Moving things

119 **Example** **Meteoric I and II** *RB*

A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

§8.8. Moving backdrops

120 **Example** **Orange Cones** *RB*

Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

§8.9. Moving the player

121 **Example** **Terror of the Sierra Madre** *RB*

Multiple player characters who take turns controlling the action.

§8.10. Removing things from play

122 **Example** **Beverage Service** *RB*

A potion that the player can drink.

123 **Example** **Spring Cleaning** *RB*

A character who sulks over objects that the player has broken (and which are now off-stage).

124 **Example** **Extra Supplies** *RB*

A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

§8.11. Now...

125 **Example** **Bee Chambers** *RB*

A maze with directions between rooms randomized at the start of play.

126 **Example** **Hatless** *RB*

It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

127 **Example** **Technological Terror** *RB*

A ray gun which destroys objects, leaving their component parts behind.

§8.15. Calling names

128 **Example** **Higher Calling** *RB*
All doors in the game automatically attempt to open if the player approaches them when they are closed.

§8.18. Randomness

129 **Example** **Do Pass Go** *RB*
A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

130 **Example** **Lanista 1** *RB*
Very simple randomized combat in which characters hit one another for a randomized amount of damage.

131 **Example** **Weathering** *RB*
The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

132 **Example** **Uptown Girls** *RB*
A stream of random pedestrians who go by the player.

§8.19. Random choices of things

133 **Example** **Candy** *RB*
One of several identical candies chosen at the start of play to be poisonous.

134 **Example** **Zork II** *RB*
A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby.

Chapter 9: Time

§9.1. When play begins

135 **Example** **Clueless** *RB*
A murderer for the mystery is selected randomly at the beginning of the game.

§9.2. Awarding points

136 **Example** **Mutt's Adventure** *RB*
Awarding points for visiting a room for the first time.

Example **No Place Like Home**

Recording a whole table of scores for specific treasures.
137 *RB*

§9.4. When play ends

138 **Example** **Big Sky Country** *RB*
Allowing the player to continue play after a fatal accident, but penalizing him by scattering his possessions around the game map.

§9.5. Every turn

139 **Example** **Witnessed 1** *RB*
A kind of battery which can be put into different devices, and which will lose power after extended use.

140 **Example** **Text Foosball** *RB*
A game of foosball which relies heavily on every-turn rules.

§9.6. The time of day

141 **Example** **IPA** *RB*
Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out if he overstays his welcome.

§9.7. Telling the time

142 **Example** **Situation Room** *RB*
Printing the time of day in 24-hour time, as in military situations.

§9.11. Future events

143 **Example** **MRE** *RB*
Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

144 **Example** **Totality** *RB*
To schedule an eclipse of the sun, which involves a number of related events.

145 **Example** **Empire** *RB*
A train which follows a schedule, stopping at a number of different locations.

146 **Example** **Hour of the Wren** *RB*
Allowing the player to make an appointment, which is then kept.

§9.12. Actions as conditions

147 **Example** **Night Sky** *RB*
A room which changes its description depending on whether an object has been examined.

148 **Example** **Zero** *RB*
A box which called "horribly heavy box" after the player has tried to take it the first time.

§9.13. The past and perfect tenses

149 **Example** **Tense Boxing** *RB*
An overview of all the variations of past and present tenses, and how they might be used.

150 **Example** **Bruneseau's Journey** *RB*
A candle which reacts to lighting and blowing actions differently depending on whether it has already been lit once.

151 **Example** **Elsie** *RB*
A door that closes automatically one turn after the player opens it.

§9.14. How many times?

152 **Example** **Infiltration** *RB*
A room whose description changes depending on the number of times the player has visited.

§9.15. How many turns?

153 **Example** **Annoyotron Jr** *RB*
A child who after a certain period in the car starts asking annoying questions.

Chapter 10: Scenes

§10.2. Creating a scene

154 **Example** **Pine 1** *RB*
Pine: Using a scene to watch for the solution of a puzzle, however arrived-at by the player.

155 **Example** **Entrapment** *RB*
A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing situation.

§10.3. Using the Scene index

156 Example **Age of Steam** *RB*
The railway-station examples so far put together into a short game called "Age of Steam".

§10.4. During scenes

157 Example **Full Moon** *RB*
Random atmospheric events which last the duration of a scene.

158 Example **Space Patrol - Stranded on Jupiter!** *RB*
We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

159 Example **Bowler Hats and Baby Geese** *RB*
Creating a category of scenes that restrict the player's behavior.

160 Example **Day One** *RB*
A scene which plays through a series of events in order, then ends when the list of events is exhausted.

§10.5. Linking scenes together

161 Example **Pine 2** *RB*
Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

§10.6. More general linkages

162 Example **The Prague Job** *RB*
Scenes used to provide pacing while the player goes through his possessions.

163 Example **Entrevaux** *RB*
Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

§10.7. Multiple beginnings and repeats

164 Example **Night and Day** *RB*
Cycling through a sequence of scenes to represent day and night following one another during a game.

165 Example **Pine 3** *RB*
Pine: Allowing the player to visit aspects of the past in memory and describe these events

to the princess, as a break from the marriage-proposal scene.

§10.8. Multiple endings

166 **ExamplePanache** *RB*
Replacing the score with a plot summary that records the events of the plot, scene by scene.

167 **ExamplePine 4** *RB*
Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the Princess has understood the point.

§10.9. Why are scenes designed this way?

168 **ExampleCheese-makers** *RB*
Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

Chapter 11: Phrases

§11.3. Pattern matching

169 **ExampleAhem** *RB*
Writing a phrase, with several variant forms, whose function is to follow a rule several times.

170 **ExampleFerragamo Again** *RB*
Using the same phrase to produce different results with different characters.

§11.5. Conditions and questions

171 **ExampleProposal** *RB*
Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

§11.7. Begin and end

172 **ExampleMatreshka** *RB*
A SEARCH [room] action that will open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

173 **ExamplePrincess and the Pea** *RB*
The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

§11.8. Otherwise

174 ★ **Example** **Numberless** *RB*
A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

§11.10. Repeat

175 ★ **Example** **Wonka's Revenge** *RB*
A lottery drum which redistributes the tickets inside whenever the player spins it.

§11.11. Repeat running through

176 ★★ **Example** **Strictly Ballroom** *RB*
People who select partners for dance lessons each turn.

§11.14. Phrase options

177 ★★ **Example** **Equipment List** *RB*
Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

§11.15. Let and temporary variables

178 ★ **Example** **M. Melmoth's Duel** *RB*
Three basic ways to inject random or not-so-random variations into text.

§11.16. New conditions, new adjectives

179 ★★★ **Example** **Owen's Law** *RB*
OUT always means "move to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

§11.17. Phrases to decide other things

180 ★ **Example** **Witnessed 2** *RB*
A piece of ghost-hunting equipment that responds depending on whether or not the meter is on and a ghost is visible or touchable from the current location.

181 ★★★ **Example** **A Haughty Spirit** *RB*
Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

§11.18. The value after and the value before

182 **Example** **Entropy** *RB*
All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

183 **Example** **The Hang of Thursdays** *RB*
Turns take a quarter day each, and the game rotates through the days of the week.

Chapter 12: Advanced Actions

§12.3. Giving instructions to other people

184 **Example** **Virtue** *RB*
Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

185 **Example** **Latris Theon** *RB*
A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

§12.4. Persuasion

186 **Example** **The Hypnotist of Blois** *RB*
A hypnotist who can make people obedient and then set them free again.

187 **Example** **Police State** *RB*
Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

§12.5. Unsuccessful attempts

188 **Example** **Generation X** *RB*
A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

§12.6. Spontaneous actions by other people

189 **Example** **IQ Test** *RB*
Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

190 **Example** **Boston Cream** *RB*
A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

§12.7. New actions

191 **Example** **Red Cross** *RB*
A DIAGNOSE command which allows the player to check on the health of someone.

192 **Example** **Frizz** *RB*
Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

193 **Example** **3 AM** *RB*
A shake command which agitates soda and makes items thump around in boxes.

§12.9. Check, carry out, report

194 **Example** **The Dark Ages Revisited** *RB*
An electric light kind of device which becomes lit when switched on and dark when switched off.

195 **Example** **Paddington** *RB*
A CUT [something] WITH [something] command which acts differently on different types of objects.

196 **Example** **Delicious, Delicious Rocks** *RB*
Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

197 **Example** **Noisemaking** *RB*
Creating a stage after the report stage of an action, during which other characters may observe and react.

§12.10. Action variables

198 **Example** **Removal** *RB*
TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

199 **Example** **Further Reasons Why All Poets Are Liars** *RB*
The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

200 **Example** **The Second Oldest Problem** *RB*
Adapting the going action so that something special can happen when going from a dark room to another dark room.

201 **Example** **Puff of Orange Smoke** *RB*
A system in which every character has a body, which is left behind when the person dies; attempts to do something to the body are redirected to the person while the person is alive.

202 **Example** **Croft** *RB*
Adding special reporting and handling for objects dropped when the player is on a supporter, and special entering rules for moving from one supporter to another.

§12.11. Making actions work for other people

203 **Example** **The Man of Steel** *RB*
An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

204 **Example** **Trying Taking Manhattan** *RB*
Replacing the inventory reporting rule with another which does something slightly different.

205 **Example** **Under Contract** *RB*
Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

§12.12. Check rules for actions by other people

206 **Example** **Get Axe** *RB*
Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

207 **Example** **Barter Barter** *RB*
Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

§12.13. Report rules for actions by other people

208 **Example** **The Man of Steel Excuses Himself** *RB*
Elaborating the report rules to be more interesting than "Clark goes west."

209 **Example** **Fate Steps In** *RB*
Fate entity which attempts to make things happen, by hook or by crook, including taking preliminary actions to set the player up a bit.

§12.15. Out of world actions

210 **Example** **Spellbreaker** *RB*
P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

211 Example **A point for never saving the game** *RB*

In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

§12.16. Reaching inside and reaching outside rules

212 Example **Carnivale** *RB*

An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

§12.17. Visible vs touchable vs carried

213 Example **Eddystone** *RB*

Creating new commands involving the standard compass directions.

214 Example **Slogar's Revenge** *RB*

Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

§12.18. Changing reachability

215 Example **Magneto's Revenge** *RB*

Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

216 Example **Waterworld** *RB*

A backdrop which the player can examine, but cannot interact with in any other way.

217 Example **Dinner is Served** *RB*

A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

§12.19. Changing visibility

218 Example **Flashlight** *RB*

Visibility set so that looking under objects produces no result unless the player has a light source to shine there (regardless of the light level of the room).

§12.20. Stored actions

Example **Bosch**

Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports. 219 RB

Example **Cactus Will Outlive Us All** 220 RB
For every character besides the player, there is an action that will cause that character to wither right up and die.

Example **Actor's Studio** 221 RB
A video camera that records actions performed in its presence, and plays them back with time-stamps.

Example **Anteaters** 222 RB
The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing actions that apply to topics, as in "look up anteater colonies in the guide".

Chapter 13: Relations

§13.2. What sentences are made up from

Example **Formal syntax of sentences** 223 RB
A more formal description of the sentence grammar used by Inform for both assertions and conditions.

§13.4. To carry, to wear, to have

Example **Celadon** 224 RB
Using the enclosure relation to let the player drop things which he only indirectly carries.

Example **Interrogation** 225 RB
A wand which, when waved, reveals the concealed items carried by people the player can see.

§13.6. Making reciprocal relations

Example **Four Cheeses** 226 RB
A system of telephones on which the player can call distant persons and have conversations.

§13.7. Relations in groups

Example **Transmutations** 227 RB
A machine that turns objects into other, similar objects.

228 ★★★ **Example** **Otranto** *RB*
A kind of rope which can be tied to objects and used to anchor the player or drag items from room to room.

§13.9. Defining new assertion verbs

229 ★ **Example** **Unthinkable Alliances** *RB*
People are to be grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give notice of quitting, and to become a lone wolf.

230 ★★★ **Example** **The Unexamined Life** *RB*
An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a nudge he wants at any given moment.

§13.10. Defining new prepositions

231 ★ **Example** **The Abolition of Love** *RB*
A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

232 ★ **Example** **Swerve left? Swerve right? Or think about it and die?** *RB*
Building a marble chute track in which a dropped marble will automatically roll downhill.

233 ★ **Example** **Beneath the Surface** *RB*
An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

234 ★★★ **Example** **Bogart** *RB*
Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

§13.11. Indirect relations

235 ★★★ **Example** **The Problem of Edith** *RB*
A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before.

§13.12. Relations which express conditions

236 ★ **Example** **Wainwright Acts** *RB*
A technical note about checking the location of door objects when characters other than the player are interacting with them.

237 ★★★ **Example** **A Humble Wayside Flower** *RB*
Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

§13.13. Relations involving values

238 ★ **Example** **Meet Market** *RB*
A case in which relations give characters multiple values of the same kind.

239 ★★★ **Example** **For Demonstration Purposes** *RB*
A character who learns new actions by watching the player performing them.

§13.14. Relations as values in their own right

240 ★ **Example** **Number Study** *RB*
The parity and joint magnitude relations explored.

§13.16. What are relations for?

241 ★★ **Example** **Murder on the Orient Express** *RB*
A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but there is physical evidence which may change their minds...

242 ★★ **Example** **What Not To Wear** *RB*
A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

243 ★★★ **Example** **Mathematical view of relations** *RB*
Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

244 ★★★ **Example** **Graph-theory view of relations** *RB*
Some notes on relations from the point of view of graph theory.

Chapter 14: Adaptive Text and Responses

§14.3. More on adapting verbs

245 ★★ **Example** **Fun with Participles** *RB*
Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on Clark's idle activity.

246 **Example Variety** *RB*
Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

247 **Example Variety 2** *RB*
This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

248 **Example Narrative Register** *RB*
Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

§14.6. Adapting demonstratives and possessives

249 **Example Olfactory Settings** *RB*
Some adaptive text for smelling the flowers, or indeed, anything else.

§14.9. Verbs as values

250 **Example History Lab** *RB*
We create phrases such as "the box we took" and "the newspaper Clark looked at" based on what has already happened in the story.

251 **Example Relevant Relations** *RB*
An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

§14.11. Changing the text of responses

252 **Example Responsive** *RB*
Altering the standard inventory text for when the player is carrying nothing.

Chapter 15: Numbers and Equations

§15.2. Numbers and real numbers

253 **Example Alias** *RB*
A telephone with phone numbers of the standard American seven-digit length.

§15.8. Units

Example rBGH

The player character's height is selected randomly at the start of play.
254 *RB*

Example **Lethal Concentration 1**
255 A poisonous gas that spreads from room to room, incapacitating or killing the player when it reaches sufficient levels. *RB*

Example **Wonderland**
256 Hiking Mount Rainier, with attention to which locations are higher and which lower than the present location. *RB*

Example **Lethal Concentration 2**
257 Poisonous gas again, only this time it sinks. *RB*

§15.12. Making the verb "to weigh"

Example **Dimensions**
258 This example draws together the previous snippets into a working implementation of the weighbridge. *RB*

Example **Lead Cuts Paper**
259 To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster. *RB*

§15.15. The parts of a number specification

Example **Zqlran Era 8**
260 Creating an alternative system of time for our game, using new units. *RB*

§15.16. Understanding specified numbers

Example **Snip**
261 A string which can be cut into arbitrary lengths, and then tied back together. *RB*

§15.17. Totals

Example **Nickel and Dimed**
262 A more intricate system of money, this time keeping track of the individual denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change. *RB*

§15.18. Equations

Example **Widget Enterprises**

Allowing the player to set a price for a widget on sale, then determining the resulting sales based on consumer demand, and the resulting profit and loss. *RB*

§15.19. Arithmetic with units

Example **Frozen Assets** *RB*
A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping.

Example **Money for Nothing** *RB*
An OFFER price FOR command, allowing the player to bargain with a flexible seller.

Example **Lemonade** *RB*
Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

Example **Savannah** *RB*
Using the liquid implementation demonstrated in Lemonade for putting out fires.

§15.20. Multiplication of units

Example **Depth** *RB*
Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

Example **Fabrication** *RB*
A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

Example **The Speed of Thought** *RB*
Describing scientifically-measured objects in units more familiar to the casual audience.

Chapter 16: Tables

§16.3. Corresponding entries

Example **Dubai** *RB*
An elevator which connects any of 27 floors in a luxury hotel.

§16.6. Repeating through tables

Example **Port Royal 4**

A cell window through which the player can see people who were in Port Royal in the current year of game-time.

272 RB

§16.9. Blank rows

273 **Example** **If It Hadn't Been For...** RB
A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

§16.10. Adding and removing rows

274 **Example** **Odyssey** RB
A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

§16.11. Sorting

275 **Example** **Jokers Wild** RB
A deck of cards which can be shuffled and dealt from.

§16.12. Listed in...

276 **Example** **Noisy Cricket** RB
Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

§16.13. Topic columns

277 **Example** **Merlin** RB
A REMEMBER command which accepts any text and looks up a response in a table of recollections.

278 **Example** **Questionable Revolutions** RB
An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

279 **Example** **The Queen of Sheba** RB
Allowing the player to use question words, and using that information to modify the response given by the other character.

§16.14. Another scoring example

280 **Example** **Goat-Cheese and Sage Chicken** RB
Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this

chapter.

§16.15. Varying which table to look at

281 **Example** **Farewell** *RB*
People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.

§16.16. Defining things with tables

282 **Example** **Sweeney** *RB*
A conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

283 **Example** **Introduction to Juggling** *RB*
Assortment of equipment defined with price and description, in a table.

§16.18. Table continuations

284 **Example** **Food Network Interactive** *RB*
Using a menu system from an extension, but adding our own material to it for this game.

§16.19. Table amendments

285 **Example** **Trieste** *RB*
Table amendment to adjust HELP commands provided for the player.

Chapter 17: Understanding

§17.1. Understand

286 **Example** **Indirection** *RB*
Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

287 **Example** **XYZZY** *RB*
Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

288 **Example** **Xylan** *RB*
Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

§17.2. New commands for old grammar

289 ★ Example **Alpaca Farm** *RB*
A generic USE action which behaves sensibly with a range of different objects.

290 ★ Example **Anchorite** *RB*
By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.

291 ★★★★ Example **Cloak of Darkness** *RB*
Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

§17.3. Overriding existing commands

292 ★ Example **The Trouble with Printing** *RB*
Making a READ command, distinct from EXAMINE, for legible objects.

293 ★★ Example **Lanista 2** *RB*
Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an ATTACK IT WITH action is created to replace regular attacking. Also folds a new DIAGNOSE command into the system.

§17.4. Standard tokens of grammar

294 ★ Example **Shawn's Bad Day** *RB*
Allowing the player to EXAMINE ALL.

295 ★★★ Example **The Left Hand of Autumn** *RB*
The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple examination command that describes groups in special ways.

§17.5. The text token

296 ★ Example **Ish.** *RB*
A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

297 ★★ Example **Nameless** *RB*
ASKing someone about an object rather than about a topic.

§17.6. Actions applying to kinds of value

298 ★ Example **Safety** RB
A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.

299 ★ Example **Tom's Midnight Garden** RB
A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running. Time on the face is also reported differently depending on whether the clock is analog or digital.

300 ★★ Example **Ibid.** RB
A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

§17.7. Understanding any, understanding rooms

301 ★ Example **One of Those Mornings** RB
A FIND command that allows the player to find a lost object anywhere

302 ★★ Example **Actaeon** RB
A FOLLOW command allowing the player to pursue a person who has just left the room.

§17.9. Understanding kinds of value

303 ★ Example **Pages** RB
A book with pages that can be read by number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

304 ★★ Example **Down in Oodville** RB
Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

305 ★★★ Example **Straw Into Gold** RB
Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or "man" -- depending on which the player last used -- until the first time the player refers to him as "Rumpelstiltskin".

§17.10. Commands consisting only of nouns

306 ★ Example **Misadventure** RB
A going by name command which does respect movement rules, and accepts names of rooms as commands.

307 **Example Safari Guide** *RB*
The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling all openable doors on the way.

§17.11. Understanding values

308 **Example Palette** *RB*
An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean") are accepted alongside everyday ones ("blue").

309 **Example Baritone, Bass** *RB*
Letting the player pick a gender (or perhaps other characteristics) before starting play.

§17.13. New tokens

310 **Example Lies** *RB*
Commands to allow the player to lie down in three different ways.

§17.15. Understanding things by their properties

311 **Example Aspect** *RB*
Understanding aspect ratios (a unit) in the names of televisions.

312 **Example Hymenaeus** *RB*
Understanding "flaming torch" and "extinguished torch" to refer to torches when lit and unlit.

313 **Example Channel 1** *RB*
Understanding channels (a number) in the names of televisions.

314 **Example Terracottissima** *RB*
The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

315 **Example Peers** *RB*
The peers of the English realm come in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince - and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

316 **Example Channel 2** *RB*
Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

317 Example **Terracottissima Maxima** *RB*
Flowerpots with textual names that might change during play.

318 Example **Tilt 1** *RB*
A deck of cards with fully implemented individual cards, which can be separately drawn and discarded, and referred to by name.

§17.16. Understanding things by their relations

319 Example **Cinco** *RB*
A taco shell that can be referred to (when it contains things) in terms of its contents.

320 Example **Puncak Jaya** *RB*
When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

321 Example **Whither?** *RB*
A door whose description says where it leads; and which automatically understands references such as "the west door" and "the east door" depending on which direction it leads from the location.

322 Example **Claims Adjustment** *RB*
An instant camera that spits out photographs of anything the player chooses to take a picture of.

§17.17. Context: understanding when

323 Example **Quiz Show** *RB*
In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

324 Example **Bibliophilia** *RB*
A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

§17.18. Changing the meaning of pronouns

325 Example **Pot of Petunias** *RB*
Responding sensibly to a pot of petunias falling from the sky.

§17.19. Does the player mean...

326 Example **Masochism Deli** *RB*
Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

§17.20. Multiple action processing

327 Example **The Best Till Last** *RB*
Reordering multiple objects for dramatic effect.

328 Example **Western Art History 305** *RB*
Allowing EXAMINE to see multiple objects with a single command.

§17.21. Understanding mistakes

329 Example **Query** *RB*
Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

330 Example **The Gorge at George** *RB*
If the player tries to TALK TO a character, suggest alternative modes of conversation.

331 Example **Hot Glass Looks Like Cold Glass** *RB*
Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

§17.22. Precedence

332 Example **Some Assembly Required** *RB*
Building different styles of shirt from component sleeves and collars.

333 Example **Lakeside Living** *RB*
Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

Chapter 18: Activities

§18.1. What are activities?

334 Example **Ant-Sensitive Sunglasses** *RB*
What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

§18.5. New activities

335 **Example** **AARP-Gnosis** *RB*
An Encyclopedia set which treats volumes in the same place as a single object, but can also be split up.

336 **Example** **Aftershock** *RB*
Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times.

337 **Example** **Crusoe** *RB*
Adding a "printing the description of something" activity.

§18.9. Deciding the concealed possessions of something

338 **Example** **Hays Code** *RB*
Clark Gable in a pin-striped suit and a pink thong.

§18.10. Printing the name of something

339 **Example** **Shipping Trunk** *RB*
A box of baking soda whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

340 **Example** **Trachypachidae Maturin 1803** *RB*
Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

341 **Example** **Chronic Hinting Syndrome** *RB*
Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

§18.11. Printing the plural name of something

342 **Example** **Hudsucker Industries** *RB*
Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

§18.12. Printing a number of something

343 **Example** **Prolegomena** *RB*
Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

§18.13. Listing contents of something

344 Example **Unpeeled** *RB*
Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

§18.15. Issuing the response text of something

345 Example **Wesponses** *RB*
Parser messages that are delivered with a speech impediment.

§18.16. Printing room description details of something

346 Example **Rules of Attraction** *RB*
A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

§18.18. Printing a refusal to act in the dark

347 Example **Zorn of Zorna** *RB*
Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

§18.22. Printing the description of a dark room

348 Example **Hohmann Transfer** *RB*
Changing the way dark rooms are described to avoid the standard Inform phrasing.

349 Example **Four Stars 1** *RB*
An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead.

§18.23. Constructing the status line

350 Example **Ways Out** *RB*
A status line that lists the available exits from the current location.

351 Example **Guided Tour** *RB*
A status line that lists the available exits from the current location, changing the names of these exits depending on whether the room has been visited or not.

§18.24. Writing a paragraph about

352 Example **Reflections** *RB*
Emphasizing the reflective quality of shiny objects whenever they are described in the

presence of the torch.

353 **Example** **Emma** *RB*
Social dynamics in which groups of people form and circulate during a party.

354 **Example** **Air Conditioning is Standard** *RB*
Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

§18.25. Listing nondescript items of something

355 **Example** **Rip Van Winkle** *RB*
A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

356 **Example** **Happy Hour** *RB*
Listing visible characters as a group, then giving some followup details in the same paragraph about specific ones.

357 **Example** **The Eye of the Idol** *RB*
A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

§18.26. Printing the locale description of something

358 **Example** **Priority Lab** *RB*
A debugging rule useful for checking the priorities of objects about to be listed.

§18.27. Choosing notable locale objects for something

359 **Example** **Low Light** *RB*
An object that is only visible and manipulable when a bright light fixture is on.

360 **Example** **Casino Banale** *RB*
Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

§18.28. Printing a locale paragraph about

361 **Example** **Kiwi** *RB*
Creating a raised supporter kind whose contents the player can't see or take from the ground.

362 Example **Copper River** RB
Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

§18.29. Deciding the scope of something

363 Example **Peeled** RB
Two different approaches to adjusting what the player can interact with, compared.

364 Example **Four Stars 2** RB
Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

365 Example **Ginger Beer** RB
A portable magic telescope which allows the player to view items in another room of his choice.

366 Example **Rock Garden** RB
A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

367 Example **Stately Gardens** RB
An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

§18.31. Asking which do you mean

368 Example **Apples** RB
Prompting the player on how to disambiguate otherwise similar objects.

369 Example **Originals** RB
Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

370 Example **Walls and Noses** RB
Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

§18.32. Supplying a missing noun/second noun

371 Example **Latin Lessons** RB
Supplying missing nouns and second nouns for other characters besides the player.

★ 372 **Example** **Minimal Movement** *RB*
Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

§18.33. Reading a command

★★ 373 **Example** **Cloves** *RB*
Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

★★ 374 **Example** **Fragment of a Greek Tragedy** *RB*
Responding to the player's input based on keywords only, and overriding the original parser entirely.

★★ 375 **Example** **North by Northwest** *RB*
Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed words.

★★★ 376 **Example** **Complimentary Peanuts** *RB*
A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

§18.34. Implicitly taking something

★ 377 **Example** **The Big Sainsbury's** *RB*
Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

★ 378 **Example** **Pizza Prince** *RB*
Providing a pizza buffet from which the player can take as many pieces as he wants.

★★★ 379 **Example** **Lollipop Guild** *RB*
Overriding the rules to allow the player to show something to another character without first taking it.

§18.35. Printing a parser error

★ 380 **Example** **WXPQ** *RB*
Creating a more sensible parser error than "that noun did not make sense in this context".

★★★ 381 **Example** **Xot** *RB*
Storing an invalid command to be repeated as text later in the game.

§18.37. Printing the banner text

382 **Example** **Bikini Atoll** *RB*
Delaying the banner for later.

§18.38. Printing the player's obituary

383 **Example** **Battle of Ridgefield** *RB*
Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

384 **Example** **Finality** *RB*
Not mentioning UNDO in the final set of options.

385 **Example** **Jamaica 1688** *RB*
Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

§18.39. Amusing a victorious player

386 **Example** **Xerxes** *RB*
Offering the player a menu of things to read after winning the game.

§18.40. Starting the virtual machine

387 **Example** **Blankness** *RB*
Emptying the status line during the first screen of the game.

Chapter 19: Rulebooks

§19.2. Named rules and rulebooks

388 **Example** **Nine AM Appointment** *RB*
A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

389 **Example** **Delayed Gratification** *RB*
A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

§19.3. New rules

Example **The Crane's Leg 2**
A description text generated based on the propensities of the player-character, following

different rulebooks for different characters.
390 *RB*

391 Example **Stone** *RB*
A soup to which the player can add ingredients, which will have different effects when the player eats.

392 Example **Bribery** *RB*
A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

§19.4. Listing rules explicitly

393 Example **Saint Eligius** *RB*
Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

394 Example **Uptempo** *RB*
Adjust time advancement so the game clock moves fifteen minutes each turn.

395 Example **Verbosity 2** *RB*
Making rooms give full descriptions each time we enter, even if we have visited before, and disallowing player use of BRIEF and SUPERBRIEF.

396 Example **Slouching** *RB*
A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

397 Example **Swigmore U.** *RB*
Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

§19.5. Changing the behaviour of rules

398 Example **Access All Areas** *RB*
The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

§19.7. The preamble of a rule

399 Example **Backus-Naur form for rules** *RB*
The full grammar Inform uses to parse rule definitions, in a standard computer-science notation.

§19.8. New rulebooks

400 ★★ **Example** **Solitude** *RB*
Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the scenery.

401 ★★★ **Example** **In Fire or in Flood** *RB*
A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

402 ★★★★ **Example** **Patient Zero** *RB*
People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

§19.9. Basis of a rulebook

403 ★ **Example** **Flotation** *RB*
Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

§19.11. Success and failure

404 ★★ **Example** **Kyoto** *RB*
Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

§19.12. Named outcomes

405 ★ **Example** **Being Peter** *RB*
A set of rules determining the attitude a character will take when asked about certain topics.

§19.13. Rulebooks producing values

406 ★ **Example** **Feline Behavior** *RB*
A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

407 ★★★ **Example** **Tilt 2** *RB*
A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

§19.15. Two rulebooks used internally

408 **Example** **Electrified** *RB*
Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances.

409 **Example** **Timeless** *RB*
A set of actions which do not take any game time at all.

410 **Example** **Endurance** *RB*
Giving different actions a range of durations using a time allotment rulebook.

411 **Example** **Escape from the Seraglio** *RB*
Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

Chapter 20: Advanced Text

§20.4. Upper and lower case letters

412 **Example** **Capital City** *RB*
To arrange that the location information normally given on the left-hand side of the status line appears in block capitals.

413 **Example** **Rocket Man** *RB*
Using case changes on any text produced by a "to say..." phrase.

§20.6. Regular expression matching

414 **Example** **Alpha** *RB*
Creating a beta-testing command that matches any line starting with punctuation.

415 **Example** **About Inform's regular expression support** *RB*
Some footnotes on Inform's regular expressions, and how they compare to those of other programming languages.

§20.7. Making new text with text substitutions

416 **Example** **Identity Theft** *RB*
Allowing the player to enter a name to be used for the player character during the game.

417 **Example** **Mirror, Mirror** *RB*
The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

418 **Example** **The Cow Exonerated** *RB*
Creating a class of matches that burn for a time and then go out, with elegant reporting when several matches go out at once.

§20.8. Replacements

419 **Example** **Blackout** *RB*
Filtering the names of rooms printed while in darkness.

420 **Example** **Fido** *RB*
A dog the player can name and un-name at will.

421 **Example** **Igpay Atinlay** *RB*
A pig Latin filter for the player's commands.

422 **Example** **Mr. Burns' Repast** *RB*
Letting the player guess types for an unidentifiable fish.

423 **Example** **Northstar** *RB*
Making Inform understand ASK JOSH TO TAKE INVENTORY as JOSH, TAKE INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

424 **Example** **Cave-troll** *RB*
Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

Chapter 21: Lists

§21.3. Saying lists of values

425 **Example** **Oyster Wide Shut** *RB*
Replacing Inform's default printing of properties such as "(closed)", "(open and providing light)", etc., with our own, more flexible variation.

§21.5. Building lists

426 **Example** **Robo 1** *RB*
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

§21.6. Lists of objects

427 Example **What Makes You Tick** *RB*
Building a fishing pole from several component parts that the player might put together in any order.

428 Example **Formicidae** *RB*
Manipulating the order in which items are handled after TAKE ALL.

§21.9. Accessing entries in a list

429 Example **Robo 2** *RB*
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

§21.10. Lengthening or shortening a list

430 Example **Leopard-skin** *RB*
A maze that the player can escape if he performs an exact sequence of actions.

431 Example **The Facts Were These** *RB*
Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

§21.11. Variations: arrays, logs, queues, stacks, sets, sieves and rings

432 Example **Circle of Misery** *RB*
Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

433 Example **Eyes, Fingers, Toes** *RB*
A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

434 Example **The Fibonacci Sequence** *RB*
The modest Leonardo Fibonacci of Pisa will be only too happy to construct his sequence on request, using an array.

435 Example **I Didn't Come All The Way From Great Portland Street** *RB*
In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

436 **Example** **Lugubrious Pete's Delicatessen** *RB*
In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

437 **Example** **Sieve of Eratosthenes** *RB*
The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

438 **Example** **Your Mother Doesn't Work Here** *RB*
Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

Chapter 22: Advanced Phrases

§22.2. Descriptions as values

439 **Example** **Curare** *RB*
A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

Chapter 23: Figures, Sounds and Files

§23.13. Writing and reading tables to external files

440 **Example** **Alien Invasion Part 23** *RB*
Keeping a preference file that could be loaded by any game in a series.

441 **Example** **Labyrinth of Ghosts** *RB*
Remembering the fates of all previous explorers of the labyrinth.

442 **Example** **Rubies** *RB*
A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

§23.14. Writing, reading and appending text to files

443 **Example** **The Fourth Body** *RB*
Notebooks in which the player can record assorted notes throughout play.

444 **Example** **The Fifth Body** *RB*
An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

§23.15. Exchanging files with other programs

445 ★★★ **Example** **Flathead News Network** *RB*
Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

Chapter 25: Releasing

§25.23. Titling and abbreviation

446 ★ **Example** **Baedeker** *RB*
Creating a floorplan of the cathedral using the locations from previous examples.

447 ★ **Example** **Port Royal 5** *RB*
Port Royal scenario given instructions for an EPS map.

448 ★ **Example** **Bay Leaves and Honey Wine** *RB*
Creating a map of Greece using the locations from previous examples.

Chapter 27: Extensions

§27.5. A simple example extension

449 ★★ **Example** **Modern Conveniences** *RB*
Exemplifying the kind of source we might use in writing extensions for kitchen and bathroom appliances.

§27.7. Extensions and story file formats

450 ★★ **Example** **Tilt 3** *RB*
Displaying the card suits from our deck of cards with red and black colored unicode symbols.

§27.15. Defining phrases in Inform 6

451 ★★★ **Example** **Pink or Blue** *RB*
Asking the player to select a gender to begin play.

§27.19. Longer extracts of Inform 6 code

452 ★ **Example** **Status line with centered text, the hard way** *RB*
A status line which has only the name of the location, centered.

§27.28. Segmented substitutions

453 **Example** **Chanel Version 1** *RB*

Making paired italic and boldface tags like those used by HTML for web pages.

§27.30. To say one of

454 **Example** **Blink** *RB*

Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

455 **Example** **Uncommon Ground** *RB*

Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

Examples in Thematic Order

Examples in Alphabetical Order
Examples in Numerical Order
General Index
Part I. Writing with Inform
Part II. The Inform Recipe Book

Chapter 1: How to Use The Recipe Book

§1.1. Preface

1 **Example** **About the examples** *WI*
An explanation of the examples in this documentation, and the asterisks attached to them. Click the heading of the example, or the example number, to reveal the text.

12 **Example** **Midsummer Day** *WI*
A few sentences laying out a garden together with some things which might be found in it.

§1.3. Disenchantment Bay

14 **Example** **Disenchantment Bay 1** *WI*
A running example in this chapter, Disenchantment Bay, involves chartering a boat. This is the first step: creating the cabin.

15 **Example** **Disenchantment Bay 2** *WI*
Disenchantment Bay: creating some of the objects in the cabin's description.

17 **Example** **Disenchantment Bay 3** *WI*
Disenchantment Bay: adding a view of the glacier.

18 **Example** **Disenchantment Bay 4** *WI*
Disenchantment Bay: fleshing out the descriptions of things on the boat.

20 **Example** **Disenchantment Bay 5** *WI*
Disenchantment Bay: adding the door and the deck to our charter boat.

23 **Example** **Disenchantment Bay 6** *WI*
Disenchantment Bay: locking up the charter boat's fishing rods.

25 **Example** **Disenchantment Bay 7** *WI*
Disenchantment Bay: making the radar and instruments switch on and off.

30 **Example** **Disenchantment Bay 9** *WI*
Disenchantment Bay: enter the charter boat's Captain.

33 **Example** **Disenchantment Bay 10** *WI*
Disenchantment Bay: things for the player and the characters to wear and carry.

34 **Example** **Disenchantment Bay 11** *WI*
Disenchantment Bay: making a holdall of the backpack.

28 **Example** **Disenchantment Bay 8** *WI*
Disenchantment Bay: a pushable chest of ice for the boat.

37 **Example** **Disenchantment Bay 12** *WI*
A final trip to Disenchantment Bay: the scenario turned into a somewhat fuller scene, with various features that have not yet been explained.

§1.4. Information Only

415 **Example** **About Inform's regular expression support** *WI*
Some footnotes on Inform's regular expressions, and how they compare to those of other programming languages.

223 **Example** **Formal syntax of sentences** *WI*
A more formal description of the sentence grammar used by Inform for both assertions and conditions.

243 **Example** **Mathematical view of relations** *WI*
Some notes on relations from a mathematical point of view, provided only to clarify some technicalities for those who are interested.

244 **Example** **Graph-theory view of relations** *WI*
Some notes on relations from the point of view of graph theory.

Example **Backus-Naur form for rules**
The full grammar Inform uses to parse rule definitions, in a standard computer-science

notation.
399 *WI*

Chapter 2: Adaptive Prose

§2.1. Varying What Is Written

71 **Example** **Fifty Ways to Leave Your Larva** *WI*
Using text substitution to make characters reply differently under the same circumstances.

169 **Example** **Ahem** *WI*
Writing a phrase, with several variant forms, whose function is to follow a rule several times.

174 **Example** **Numberless** *WI*
A simple exercise in printing the names of random numbers, comparing the use of "otherwise if...", a switch statement, or a table-based alternative.

178 **Example** **M. Melmoth's Duel** *WI*
Three basic ways to inject random or not-so-random variations into text.

249 **Example** **Olfactory Settings** *WI*
Some adaptive text for smelling the flowers, or indeed, anything else.

252 **Example** **Responsive** *WI*
Altering the standard inventory text for when the player is carrying nothing.

343 **Example** **Prolegomena** *WI*
Replacing precise numbers with "some" or other quantifiers when too many objects are clustered together for the player to count at a glance.

345 **Example** **Wesponses** *WI*
Parser messages that are delivered with a speech impediment.

413 **Example** **Rocket Man** *WI*
Using case changes on any text produced by a "to say..." phrase.

419 **Example** **Blackout** *WI*
Filtering the names of rooms printed while in darkness.

439 **Example** **Curare** *WI*
A phrase that chooses and names the least-recently selected item from the collection given, allowing the text to cycle semi-randomly through a group of objects.

454 **Example** **Blink** *WI*
Making a "by atmosphere" token, allowing us to design our own text variations such as "[one of]normal[or]gloomy[or]scary[by atmosphere]".

245 **Example** **Fun with Participles** *WI*
Creating dynamic room descriptions that contain sentences such as "Clark is here, wasting time" or "Clark is here, looking around" depending on Clark's idle activity.

246 **Example** **Variety** *WI*
Suppose we want all of our action responses to display some randomized variety. We could do this by laboriously rewriting all of the response texts, but this example demonstrates an alternative.

247 **Example** **Variety 2** *WI*
This builds on the Variety example to add responses such as "You are now carrying the fedora" that describe relations that result from a given verb, as alternate responses.

250 **Example** **History Lab** *WI*
We create phrases such as "the box we took" and "the newspaper Clark looked at" based on what has already happened in the story.

251 **Example** **Relevant Relations** *WI*
An example of how to create room descriptions that acknowledge particular relations using their assigned verbs, rather than by the heavily special-cased code used by the standard library.

60 **Example** **Ballpark** *WI*
A new "to say" definition which allows the author to say "[a number in round numbers]" and get verbal descriptions like "a couple of" or "a few" as a result.

72 **Example** **Fifty Times Fifty Ways** *WI*
Writing your own rules for how to carry out substitutions.

248 **Example** **Narrative Register** *WI*
Suppose we want all of our action responses to vary depending on some alterable quality of the narrator, so that sometimes they're slangy, sometimes pompous or archaic.

305 **Example** **Straw Into Gold** *WI*
Creating a Rumpelstiltskin character who is always referred to as "dwarf", "guy", "dude", or "man" -- depending on which the player last used -- until the first time the player refers to

him as "Rumpelstiltskin".

§2.2. Varying What Is Read

11 **Example** **First Name Basis** *WI*
Allowing the player to use different synonyms to refer to something.

323 **Example** **Quiz Show** *WI*
In this example by Mike Tarbert, the player can occasionally be quizzed on random data from a table; the potential answers will only be understood if a question has just been asked.

325 **Example** **Pot of Petunias** *WI*
Responding sensibly to a pot of petunias falling from the sky.

19 **Example** **Laura** *WI*
Some general advice about creating objects with unusual or awkward names, and a discussion of the use of printed names.

43 **Example** **Vouvray** *WI*
Adding synonyms to an entire kind of thing.

375 **Example** **North by Northwest** *WI*
Creating additional compass directions between those that already exist (for instance, NNW) -- and dealing with an awkwardness that arises when the player tries to type "north-northwest". The example demonstrates a way around the nine-character limit on parsed words.

§2.3. Using the Player's Input

421 **Example** **Igpay Atinlay** *WI*
A pig Latin filter for the player's commands.

422 **Example** **Mr. Burns' Repast** *WI*
Letting the player guess types for an unidentifiable fish.

317 **Example** **Terracottissima Maxima** *WI*
Flowerpots with textual names that might change during play.

381 **Example** **Xot** *WI*
Storing an invalid command to be repeated as text later in the game.

Chapter 3: Place

§3.1. Room Descriptions

147 **Example** **Night Sky** *WI*
A room which changes its description depending on whether an object has been examined.

152 **Example** **Infiltration** *WI*
A room whose description changes depending on the number of times the player has visited.

334 **Example** **Ant-Sensitive Sunglasses** *WI*
What are activities good for? Controlling output when we want the same action to be able to produce very flexible text depending on the state of the world -- in this case, making highly variable room description and object description text.

355 **Example** **Rip Van Winkle** *WI*
A simple way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description.

358 **Example** **Priority Lab** *WI*
A debugging rule useful for checking the priorities of objects about to be listed.

359 **Example** **Low Light** *WI*
An object that is only visible and manipulable when a bright light fixture is on.

4 **Example** **Slightly Wrong** *WI*
A room whose description changes slightly after our first visit there.

357 **Example** **The Eye of the Idol** *WI*
A systematic way to allow objects in certain places to be described in the room description body text rather than in paragraphs following the room description, and to control whether supporters list their contents or not.

362 **Example** **Copper River** *WI*
Manipulating room descriptions so that only interesting items are mentioned, while objects that are present but not currently useful to the player are ignored.

§3.2. Map

5 **Example** **Port Royal 1** *WI*
A partial implementation of Port Royal, Jamaica, set before the earthquake of 1692 demolished large portions of the city.

8 **Example** **Port Royal 2** *WI*
Another part of Port Royal, with less typical map connections.

10 **Example** **Port Royal 3** *WI*
Division of Port Royal into regions.

78 **Example** **All Roads Lead to Mars** *WI*
Layout where the player is allowed to wander any direction he likes, and the map will arrange itself in order so that he finds the correct "next" location.

125 **Example** **Bee Chambers** *WI*
A maze with directions between rooms randomized at the start of play.

134 **Example** **Zork II** *WI*
A "Carousel Room", as in Zork II, where moving in any direction from the room leads (at random) to one of the eight rooms nearby.

286 **Example** **Indirection** *WI*
Renaming the directions of the compass so that "white" corresponds to north, "red" to east, "yellow" to south, and "black" to west.

40 **Example** **Prisoner's Dilemma** *WI*
A button that causes a previously non-existent exit to come into being.

41 **Example** **The World of Charles S. Roberts** *WI*
Replacing the ordinary compass bearings with a set of six directions to impose a hexagonal rather than square grid on the landscape.

101 **Example** **A&E** *WI*
Using regions to block access to an entire area when the player does not carry a pass, regardless of which entrance he uses.

§3.3. Position Within Rooms

199 **Example** **Further Reasons Why All Poets Are Liars** *WI*
The young William Wordsworth, pushing a box about in his room, must struggle to achieve a Romantic point of view.

7 **Example** **Starry Void** *WI*
Creating a booth that can be seen from the outside, opened and closed, and entered as a separate room.

§3.4. Continuous Spaces and The Outdoors

216 **Example** **Waterworld** *WI*
A backdrop which the player can examine, but cannot interact with in any other way.

62 **Example** **Tiny Garden** *WI*
A lawn made up of several rooms, with part of the description written automatically.

79 **Example** **Hotel Stechelberg** *WI*
Signposts such as those provided on hiking paths in the Swiss Alps, which show the correct direction and hiking time to all other locations.

212 **Example** **Carnivale** *WI*
An alternative to backdrops when we want something to be visible from a distance but only touchable from one room.

213 **Example** **Eddystone** *WI*
Creating new commands involving the standard compass directions.

366 **Example** **Rock Garden** *WI*
A simple open landscape where the player can see between rooms and will automatically move to touch things in distant rooms.

80 **Example** **A View of Green Hills** *WI*
A LOOK [direction] command which allows the player to see descriptions of the nearby landscape.

367 **Example** **Stately Gardens** *WI*
An open landscape where the player can see landmarks in nearby areas, with somewhat more complex room descriptions than the previous example, and in which we also account for size differences between things seen at a distance.

§3.5. Doors, Staircases, and Bridges

45 **Example** **Something Narsty** *WI*
A staircase always open and never openable.

63 **Example** **When?** *WI*
A door whose description says "...leads east" in one place and "...leads west" in the other.

89 **Example** **Hayseed** *WI*
A refinement of our staircase kind which can be climbed.

128 **Example** **Higher Calling** *WI*
All doors in the game automatically attempt to open if the player approaches them when they are closed.

236 **Example** **Wainwright Acts** *WI*
A technical note about checking the location of door objects when characters other than the player are interacting with them.

321 **Example** **Whither?** *WI*
A door whose description says where it leads; and which automatically understands references such as "the west door" and "the east door" depending on which direction it leads from the location.

24 **Example** **Neighborhood Watch** *WI*
A locked door that can be locked or unlocked without a key from one side, but not from the other.

106 **Example** **One Short Plank** *WI*
A plank bridge which breaks if the player is carrying something when he goes across it. Pushing anything over the bridge is forbidden outright.

151 **Example** **Elsie** *WI*
A door that closes automatically one turn after the player opens it.

22 **Example** **Garibaldi 1** *WI*
Providing a security readout device by which the player can check on the status of all doors in the game.

64 **Example** **Whence?** *WI*
A kind of door that always automatically describes the direction it opens and what lies on the far side (if that other room has been visited).

§3.6. Windows

116 **Example** **Vitrine** *WI*
An electrochromic window that becomes transparent or opaque depending on whether it is currently turned on.

21 **Example** **Escape** *WI*
Window that can be climbed through or looked through.

217 **Example** **Dinner is Served** *WI*
A window between two locations. When the window is open, the player can reach through into the other location; when it isn't, access is barred.

272 **Example** **Port Royal 4** *WI*
A cell window through which the player can see people who were in Port Royal in the current year of game-time.

181 **Example** **A Haughty Spirit** *WI*
Windows overlooking lower spaces which will prevent the player from climbing through if the lower space is too far below.

§3.7. Lighting

194 **Example** **The Dark Ages Revisited** *WI*
An electric light kind of device which becomes lit when switched on and dark when switched off.

312 **Example** **Hymenaeus** *WI*
Understanding "flaming torch" and "extinguished torch" to refer to torches when lit and unlit.

352 **Example** **Reflections** *WI*
Emphasizing the reflective quality of shiny objects whenever they are described in the presence of the torch.

363 **Example** **Peeled** *WI*
Two different approaches to adjusting what the player can interact with, compared.

26 **Example** **Down Below** *WI*
A light switch which makes the room it is in dark or light.

51 **Example** **The Undertomb 2** *WI*
Flickering lantern-light effects added to the Undertomb.

348 **Example** **Hohmann Transfer** *WI*
Changing the way dark rooms are described to avoid the standard Inform phrasing.

81 **Example** **Unblinking** *WI*
Finding a best route through light-filled rooms only, leaving aside any that might be dark.

347 **Example** **Zorn of Zorna** *WI*
Light levels vary depending on the number of candles the player has lit, and this determines whether or not he is able to examine detailed objects successfully.

Example **Four Stars 1**

An elaboration of the idea that when light is absent, the player should be given a description of what he can smell and hear, instead. 349 WI

Example **Cloak of Darkness** 291 WI
Implementation of "Cloak of Darkness", a simple example game that for years has been used to demonstrate the features of IF languages.

§3.8. Sounds

Example **The Undertomb 1** 50 WI
A small map of dead ends, in which the sound of an underground river has different strengths in different caves.

Example **Four Stars 2** 364 WI
Using "deciding the scope" to change the content of lists such as "the list of audible things which can be touched by the player".

Example **The Art of Noise** 95 WI
Things are all assigned their own noise (or silence). Listening to the room in general reports on all the things that are currently audible.

§3.9. Passers-By, Weather and Astronomical Events

Example **Weathering** 131 WI
The automatic weather station atop Mt. Pisgah shows randomly fluctuating temperature, pressure and cloud cover.

Example **Full Moon** 157 WI
Random atmospheric events which last the duration of a scene.

Example **Night and Day** 164 WI
Cycling through a sequence of scenes to represent day and night following one another during a game.

Example **Totality** 144 WI
To schedule an eclipse of the sun, which involves a number of related events.

Example **Orange Cones** 120 WI
Creating a traffic backdrop that appears in all road rooms except the one in which the player has laid down orange cones.

Example **Uptown Girls** 132 WI
A stream of random pedestrians who go by the player.

Chapter 4: Time and Plot

§4.1. The Passage Of Time

142 **Example** **Situation Room** *WI*
Printing the time of day in 24-hour time, as in military situations.

377 **Example** **The Big Sainsbury's** *WI*
Making implicit takes add a minute to the clock, just as though the player had typed TAKE THING explicitly.

394 **Example** **Uptempo** *WI*
Adjust time advancement so the game clock moves fifteen minutes each turn.

409 **Example** **Timeless** *WI*
A set of actions which do not take any game time at all.

410 **Example** **Endurance** *WI*
Giving different actions a range of durations using a time allotment rulebook.

183 **Example** **The Hang of Thursdays** *WI*
Turns take a quarter day each, and the game rotates through the days of the week.

260 **Example** **Zqlran Era 8** *WI*
Creating an alternative system of time for our game, using new units.

§4.2. Scripted Scenes

162 **Example** **The Prague Job** *WI*
Scenes used to provide pacing while the player goes through his possessions.

155 **Example** **Entrapment** *WI*
A scene in which the player is allowed to explore as much as he likes, but another character strolls in as soon as he has gotten himself into an awkward or embarrassing situation.

159 **Example** **Bowler Hats and Baby Geese** *WI*
Creating a category of scenes that restrict the player's behavior.

Example **Day One**

A scene which plays through a series of events in order, then ends when the list of events is exhausted.

160 WI

§4.3. Event Scheduling

141 **Example** **IPA** WI
Shops which each have opening and closing hours, so that it is impossible to go in at the wrong times, and the player is kicked out if he overstays his welcome.

146 **Example** **Hour of the Wren** WI
Allowing the player to make an appointment, which is then kept.

354 **Example** **Air Conditioning is Standard** WI
Uses "writing a paragraph about" to make person and object descriptions that vary considerably depending on what else is going on in the room, including some randomized NPC interactions with objects or with each other.

§4.4. Scene Changes

156 **Example** **Age of Steam** WI
The railway-station examples so far put together into a short game called "Age of Steam".

119 **Example** **Meteoric I and II** WI
A meteor in the night sky which is visible from many rooms, so needs to be a backdrop, but which does not appear until 11:31 PM.

158 **Example** **Space Patrol - Stranded on Jupiter!** WI
We'll be back in just a moment, with more exciting adventures of the... Space Patrol!

163 **Example** **Entrevaux** WI
Organizing the game by scenes, where each scene has a location and prop lists so that it can be set up automatically.

§4.5. Flashbacks

165 **Example** **Pine 3** WI
Pine: Allowing the player to visit aspects of the past in memory and describe these events to the princess, as a break from the marriage-proposal scene.

167 **Example** **Pine 4** WI
Pine: Adding a flashback scene that, instead of repeating endlessly, repeats only until the Princess has understood the point.

§4.6. Plot Management

209 **Example** **Fate Steps In** *WI*
Fate entity which attempts to make things happen, by hook or by crook, including taking preliminary actions to set the player up a bit.

Chapter 5: The Viewpoint Character

§5.1. The Human Body

254 **Example** **rBGH** *WI*
The player character's height is selected randomly at the start of play.

396 **Example** **Slouching** *WI*
A system of postures allowing the player and other characters to sit, stand, or lie down explicitly or implicitly on a variety of enterable supporters or containers, or in location.

59 **Example** **The Night Before** *WI*
Instructing Inform to prefer different interpretations of EXAMINE NOSE, depending on whether the player is alone, in company, or with Rudolph the Red-nosed Reindeer.

§5.2. Traits Determined By the Player

416 **Example** **Identity Theft** *WI*
Allowing the player to enter a name to be used for the player character during the game.

309 **Example** **Baritone, Bass** *WI*
Letting the player pick a gender (or perhaps other characteristics) before starting play.

451 **Example** **Pink or Blue** *WI*
Asking the player to select a gender to begin play.

§5.3. Characterization

73 **Example** **Finishing School** *WI*
The "another" adjective for rules such as "in the presence of another person".

85 **Example** **Bad Hair Day** *WI*
Change the player's appearance in response to EXAMINE ME.

Example **Dearth and the Maiden**
Our heroine, fallen among gentleman highwaymen, is restrained by her own modesty and

seemliness.
109 *WI*

§5.4. Background

277 **Example** **Merlin** *WI*
A REMEMBER command which accepts any text and looks up a response in a table of recollections.

301 **Example** **One of Those Mornings** *WI*
A FIND command that allows the player to find a lost object anywhere

§5.5. Memory and Knowledge

149 **Example** **Tense Boxing** *WI*
An overview of all the variations of past and present tenses, and how they might be used.

320 **Example** **Puncak Jaya** *WI*
When a character is not visible, responding to such commands as EXAMINE PETER and PETER, HELLO with a short note that the person in question is no longer visible.

148 **Example** **Zero** *WI*
A box which called "horribly heavy box" after the player has tried to take it the first time.

360 **Example** **Casino Banale** *WI*
Creating room descriptions and object descriptions that change as the player learns new facts and pieces things together.

§5.6. Viewpoint

390 **Example** **The Crane's Leg 2** *WI*
A description text generated based on the propensities of the player-character, following different rulebooks for different characters.

455 **Example** **Uncommon Ground** *WI*
Making a "by viewpoint" token, allowing us to design our own text variations such as "[show to yourself]quaint[to Lolita]thrilling[to everyone else]squalid[end show]" depending on the identity of the player at the moment.

52 **Example** **The Crane's Leg 1** *WI*
A description text that automatically highlights the ways in which the object differs from a standard member of its kind.

121 ★★★ **Example** **Terror of the Sierra Madre** *WI*
Multiple player characters who take turns controlling the action.

Chapter 6: Commands

§6.3. Modifying Existing Commands

214 ★★★ **Example** **Slogar's Revenge** *WI*
Creating an amulet of tumblers that can be used to lock and unlock things even when it is worn, overriding the usual requirement that keys be carried.

§6.4. Looking

3 ★ **Example** **Verbosity 1** *WI*
Making rooms give brief room descriptions when revisited.

395 ★ **Example** **Verbosity 2** *WI*
Making rooms give full descriptions each time we enter, even if we have visited before, and disallowing player use of BRIEF and SUPERBRIEF.

§6.5. Examining

44 ★ **Example** **Odin** *WI*
Replacing "You see nothing special..." with a different default message for looking at something nondescript.

68 ★★ **Example** **Beekeeper's Apprentice** *WI*
Making the SEARCH command examine all the scenery in the current location.

295 ★★★ **Example** **The Left Hand of Autumn** *WI*
The possibility of using a [things] token opens up some interesting complications, because we may want actions on multiple items to be reported differently from actions on just one. Here we look at how to make a multiple examination command that describes groups in special ways.

337 ★★★ **Example** **Crusoe** *WI*
Adding a "printing the description of something" activity.

§6.6. Looking Under and Hiding

98 ★ **Example** **Beachfront** *WI*
An item that the player can't interact with until he has found it by searching the scenery.

172 **Example Matreshka** *WI*
A SEARCH [room] action that will open every container the player can see, stopping only when there don't remain any that are closed, unlocked, and openable.

218 **Example Flashlight** *WI*
Visibility set so that looking under objects produces no result unless the player has a light source to shine there (regardless of the light level of the room).

233 **Example Beneath the Surface** *WI*
An "underlying" relation which adds to the world model the idea of objects hidden under other objects.

§6.7. Inventory

425 **Example Oyster Wide Shut** *WI*
Replacing Inform's default printing of properties such as "(closed)", "(open and providing light)", etc., with our own, more flexible variation.

177 **Example Equipment List** *WI*
Overview of all the phrase options associated with listing, and examples of how to change the inventory list into some other standard formats.

65 **Example Persephone** *WI*
Separate the player's inventory listing into two parts, so that it says "you are carrying..." and then (if the player is wearing anything) "You are also wearing...".

204 **Example Trying Taking Manhattan** *WI*
Replacing the inventory reporting rule with another which does something slightly different.

§6.8. Taking, Dropping, Inserting and Putting

16 **Example Replanting** *WI*
Changing the response when the player tries to take something that is scenery.

90 **Example Morning After** *WI*
When the player picks something up which he hasn't already examined, the object is described.

198 **Example Removal** *WI*
TAKE expanded to give responses such as "You take the book from the shelf." or "You pick up the toy from the ground."

Example Celadon
Using the enclosure relation to let the player drop things which he only indirectly carries.

86 ★★★ Example **Democratic Process** WI
Make PUT and INSERT commands automatically take objects if the player is not holding them.

202 ★★★ Example **Croft** WI
Adding special reporting and handling for objects dropped when the player is on a supporter, and special entering rules for moving from one supporter to another.

379 ★★★ Example **Lollipop Guild** WI
Overriding the rules to allow the player to show something to another character without first taking it.

87 ★★★★ Example **Sand** WI
Extend PUT and INSERT handling to cases where multiple objects are intended at once.

§6.9. Going, Pushing Things in Directions

100 ★ Example **Veronica** WI
An effect that occurs only when the player leaves a region entirely.

105 ★ Example **Mattress King** WI
Adding extra phrasing to the action to PUSH something in a direction.

200 ★ Example **The Second Oldest Problem** WI
Adapting the going action so that something special can happen when going from a dark room to another dark room.

306 ★ Example **Misadventure** WI
A going by name command which does respect movement rules, and accepts names of rooms as commands.

372 ★ Example **Minimal Movement** WI
Supplying a default direction for "go", so that "leave", "go", etc., are always interpreted as "out".

393 ★ Example **Saint Eligius** WI
Adding a first look rule that comments on locations when we visit them for the first time, inserting text after objects are listed but before any "every turn" rules might occur.

6 Example **Up and Up** *WI*
Adding a short message as the player approaches a room, before the room description itself appears.

256 Example **Wonderland** *WI*
Hiking Mount Rainier, with attention to which locations are higher and which lower than the present location.

307 Example **Safari Guide** *WI*
The same functionality, but making the player continue to move until he reaches his destination or a barrier, handling all openable doors on the way.

102 Example **Bumping into Walls** *WI*
Offering the player a list of valid directions if he tries to go in a direction that leads nowhere.

103 Example **Polarity** *WI*
A "go back" command that keeps track of the direction from which the player came, and sends him back.

107 Example **Provenance Unknown** *WI*
Allowing something like PUSH TELEVISION EAST to push the cart on which the television rests.

108 Example **Zorb** *WI*
Replacing the message the player receives when attempting to push something that isn't pushable, and also to remove the restriction that objects cannot be pushed up or down.

179 Example **Owen's Law** *WI*
OUT always means "move to an outdoors room, or else to a room with more exits than this one has"; IN always means the opposite.

§6.10. Entering and Exiting, Sitting and Standing

206 Example **Get Axe** *WI*
Changing the check rules to try automatically leaving a container before attempting to take it. (And arranging things so that other people will do likewise.)

290 Example **Anchorite** *WI*
By default, Inform understands GET OFF, GET UP, or GET OUT when the player is sitting or standing on an enterable object. We might also want to add GET DOWN and DOWN as exit commands, though.

310 Example **Lies** *WI*
Commands to allow the player to lie down in three different ways.

§6.11. Waiting, Sleeping

388 **Example** **Nine AM Appointment** *WI*
A WAIT [number] MINUTES command which advances through an arbitrary number of turns.

47 **Example** **Change of Basis** *WI*
Implementing sleeping and wakeful states.

389 **Example** **Delayed Gratification** *WI*
A WAIT UNTIL [time] command which advances until the game clock reaches the correct hour.

§6.13. Magic Words

287 **Example** **XYZZY** *WI*
Basics of adding a new command reviewed, for the case of the simple magic word XYZZY.

§6.14. Remembering, Converting and Combining Actions

88 **Example** **Fine Laid** *WI*
Making writing that can be separately examined from the paper on which it appears, but which directs all other actions to the paper.

92 **Example** **Lucy** *WI*
Redirecting a question about one topic to ask about another.

220 **Example** **Cactus Will Outlive Us All** *WI*
For every character besides the player, there is an action that will cause that character to wither right up and die.

430 **Example** **Leopard-skin** *WI*
A maze that the player can escape if he performs an exact sequence of actions.

435 **Example** **I Didn't Come All The Way From Great Portland Street** *WI*
In this fiendishly difficult puzzle, which may perhaps owe some inspiration to a certain BBC Radio panel game (1967-), a list is used as a set of actions to help enforce the rule that the player must keep going for ten turns without hesitation, repetition, or deviating from the subject on the card.

222 **Example** **Anteaters** *WI*
The player carries a gizmo that is able to record actions performed by the player, then force him to repeat them when the gizmo is dropped. This includes storing actions that apply to topics, as in "look up anteater colonies in the guide".

§6.15. Actions on Multiple Objects

294 Example **Shawn's Bad Day** *WI*
Allowing the player to EXAMINE ALL.

327 Example **The Best Till Last** *WI*
Reordering multiple objects for dramatic effect.

328 Example **Western Art History 305** *WI*
Allowing EXAMINE to see multiple objects with a single command.

411 Example **Escape from the Seraglio** *WI*
Replacing the usual response to TAKE ALL so that instead of output such as "grapes: Taken. orange: Taken.", Inform produces variable responses in place of "grapes:".

428 Example **Formicidae** *WI*
Manipulating the order in which items are handled after TAKE ALL.

431 Example **The Facts Were These** *WI*
Creating a variant GIVE action that lets the player give multiple objects simultaneously with commands like GIVE ALL TO ATTENDANT or GIVE THREE DOLLARS TO ATTENDANT or GIVE PIE AND HAT TO ATTENDANT. The attendant accepts the gifts only if their total combined value matches some minimum amount.

§6.17. Clarification and Correction

289 Example **Alpaca Farm** *WI*
A generic USE action which behaves sensibly with a range of different objects.

368 Example **Apples** *WI*
Prompting the player on how to disambiguate otherwise similar objects.

380 Example **WXPQ** *WI*
Creating a more sensible parser error than "that noun did not make sense in this context".

370 Example **Walls and Noses** *WI*
Responding to "EXAMINE WALL" with "In which direction?", and to "EXAMINE NOSE" with "Whose nose do you mean, Frederica's, Betty's, Wilma's or your own?"

424 Example **Cave-troll** *WI*
Determining that the command the player typed is invalid, editing it, and re-examining it to see whether it now reads correctly.

§6.18. Alternatives To Standard Parsing

304 **Example** **Down in Oodville** *WI*
Offering the player a choice of numbered options at certain times, without otherwise interfering with his ability to give regular commands.

373 **Example** **Cloves** *WI*
Accepting adverbs anywhere in a command, registering what the player typed but then cutting them out before interpreting the command.

374 **Example** **Fragment of a Greek Tragedy** *WI*
Responding to the player's input based on keywords only, and overriding the original parser entirely.

Chapter 7: Other Characters

§7.1. Getting Acquainted

31 **Example** **Belfry** *WI*
You can see a bat, a bell, some woodworm, William Snelson, the sexton's wife, a bellringer and your local vicar here.

135 **Example** **Clueless** *WI*
A murderer for the mystery is selected randomly at the beginning of the game.

238 **Example** **Meet Market** *WI*
A case in which relations give characters multiple values of the same kind.

32 **Example** **Gopher-wood** *WI*
Changing the name of a character in the middle of play, removing the article.

315 **Example** **Peers** *WI*
The peers of the English realm come in six flavours - Baron, Viscount, Earl, Marquess, Duke and Prince - and must always be addressed properly. While a peerage is for life, it may at the royal pleasure be promoted.

237 **Example** **A Humble Wayside Flower** *WI*
Relations track the relationships between one character and another. Whenever the player meets a relative of someone he already knows, he receives a brief introduction.

§7.2. Liveliness

Example **Annoyotron Jr**

A child who after a certain period in the car starts asking annoying questions.
153 *WI*

Example **Camp Bethel** **67** *WI*
Creating characters who change their behavior from turn to turn, and a survey of other common uses for alternative texts.

Example **Lean and Hungry** **76** *WI*
A thief who will identify and take any valuable thing lying around that he is able to touch.

Example **Text Foosball** **140** *WI*
A game of foosball which relies heavily on every-turn rules.

§7.3. Reactive Characters

Example **Zodiac** **96** *WI*
Several variations on "doing something other than...", demonstrating different degrees of restriction.

Example **Pine 1** **154** *WI*
Pine: Using a scene to watch for the solution of a puzzle, however arrived-at by the player.

Example **Police State** **187** *WI*
Several friends who obey you; a policeman who doesn't (but who takes a dim view of certain kinds of antics).

Example **Search and Seizure** **38** *WI*
A smuggler who has items, some of which are hidden.

Example **Noisemaking** **197** *WI*
Creating a stage after the report stage of an action, during which other characters may observe and react.

Example **Revenge of the Fussy Table** **82** *WI*
A small game about resentful furniture and inconvenient objects.

Example **A Day For Fresh Sushi** **112** *WI*
A complete story by Emily Short, called "A Day for Fresh Sushi", rewritten using Inform 7. Noteworthy is the snarky commenter who remarks on everything the player does, but only the first time each action is performed.

§7.4. Barter and Exchange

392 **Example Bribery** *WI*
A GIVE command that gets rid of Inform's default refusal message in favor of something a bit more sophisticated.

207 **Example Barter Barter** *WI*
Allowing characters other than the player to give objects to one another, accounting for the possibility that some items may not be desired by the intended recipients.

§7.5. Combat and Death

130 **Example Lanista 1** *WI*
Very simple randomized combat in which characters hit one another for a randomized amount of damage.

191 **Example Red Cross** *WI*
A DIAGNOSE command which allows the player to check on the health of someone.

201 **Example Puff of Orange Smoke** *WI*
A system in which every character has a body, which is left behind when the person dies; attempts to do something to the body are redirected to the person while the person is alive.

293 **Example Lanista 2** *WI*
Randomized combat in which the damage done depends on what weapons the characters are wielding, and in which an ATTACK IT WITH action is created to replace regular attacking. Also folds a new DIAGNOSE command into the system.

113 **Example Don Pedro's Revenge** *WI*
Combat scenario in which the player's footing and position changes from move to move, and the command prompt also changes to reflect that.

127 **Example Technological Terror** *WI*
A ray gun which destroys objects, leaving their component parts behind.

§7.6. Getting Started with Conversation

330 **Example The Gorge at George** *WI*
If the player tries to TALK TO a character, suggest alternative modes of conversation.

110 **Example Mimicry** *WI*
People who must be greeted before conversation can begin.

§7.7. Saying Simple Things

91 **Example Sybil 1** *WI*
Direct all ASK, TELL, and ANSWER commands to ASK, and accept multiple words for certain cases.

93 **Example Sybil 2** *WI*
Making the character understand YES, SAY YES TO CHARACTER, TELL CHARACTER YES, ANSWER YES, and CHARACTER, YES.

171 **Example Proposal** *WI*
Asking the player a yes/no question which he must answer, and another which he may answer or not as he chooses.

297 **Example Nameless** *WI*
ASKing someone about an object rather than about a topic.

§7.8. Saying Complicated Things

281 **Example Farewell** *WI*
People who respond to conversational gambits, summarize what they said before if asked again, and provide recap of conversation that is past.

282 **Example Sweeney** *WI*
A conversation where each topic may have multiple questions and answers associated with it, and where a given exchange can lead to new additions to the list.

168 **Example Cheese-makers** *WI*
Scenes used to control the way a character reacts to conversation and comments, using a TALK TO command.

376 **Example Complimentary Peanuts** *WI*
A character who responds to keywords in the player's instructions and remarks, even if there are other words included.

§7.10. Character Emotion

405 **Example Being Peter** *WI*
A set of rules determining the attitude a character will take when asked about certain topics.

170 **Example Ferragamo Again** *WI*
Using the same phrase to produce different results with different characters.

§7.11. Character Knowledge and Reasoning

241 **Example** **Murder on the Orient Express** *WI*
A number of sleuths (the player among them) find themselves aboard the Orient Express, where a murder has taken place, and one of them is apparently the culprit. Naturally they do not agree on whom, but there is physical evidence which may change their minds...

235 **Example** **The Problem of Edith** *WI*
A conversation in which the main character tries to build logical connections between what the player is saying now and what went immediately before.

278 **Example** **Questionable Revolutions** *WI*
An expansion on the previous idea, only this time we store information and let characters answer depending on their expertise in a given area.

279 **Example** **The Queen of Sheba** *WI*
Allowing the player to use question words, and using that information to modify the response given by the other character.

341 **Example** **Chronic Hinting Syndrome** *WI*
Using name-printing rules to keep track of whether the player knows about objects, and also to highlight things he might want to follow up.

§7.12. Characters Following a Script

426 **Example** **Robo 1** *WI*
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

438 **Example** **Your Mother Doesn't Work Here** *WI*
Your hard-working mother uses a list as a stack: urgent tasks are added to the end of the list, interrupting longer-term plans.

161 **Example** **Pine 2** *WI*
Pine: Adding a conversation with the princess, in which a basic set of facts must be covered before the scene is allowed to end.

429 **Example** **Robo 2** *WI*
A robot which watches and records the player's actions, then tries to repeat them back in the same order when he is switched into play-back mode.

§7.13. Traveling Characters

77 **Example** **Mistress of Animals** *WI*
A person who moves randomly between rooms of the map.

39 **Example** **Van Helsing** *WI*
A character who approaches the player, then follows him from room to room.

274 **Example** **Odyssey** *WI*
A person who follows a path predetermined and stored in a table, and who can be delayed if the player tries to interact with her.

302 **Example** **Actaeon** *WI*
A FOLLOW command allowing the player to pursue a person who has just left the room.

185 **Example** **Latris Theon** *WI*
A person who can accept instructions to go to new destinations and move towards them according to the most reasonable path.

402 **Example** **Patient Zero** *WI*
People who wander around the map performing various errands, and in the process spread a disease which only the player can eradicate.

§7.14. Obedient Characters

184 **Example** **Virtue** *WI*
Defining certain kinds of behavior as inappropriate, so that other characters will refuse indignantly to do any such thing.

186 **Example** **The Hypnotist of Blois** *WI*
A hypnotist who can make people obedient and then set them free again.

371 **Example** **Latin Lessons** *WI*
Supplying missing nouns and second nouns for other characters besides the player.

188 **Example** **Generation X** *WI*
A person who goes along with the player's instructions, but reluctantly, and will get annoyed after too many repetitions of the same kind of unsuccessful command.

423 **Example** **Northstar** *WI*
Making Inform understand ASK JOSH TO TAKE INVENTORY as JOSH, TAKE INVENTORY. This requires us to use a regular expression on the player's command, replacing some of the content.

239 **Example** **For Demonstration Purposes** *WI*
A character who learns new actions by watching the player performing them.

205 **Example** **Under Contract** *WI*
Creating a person who accepts most instructions and reacts correctly when a request leads implicitly to inappropriate behavior.

§7.15. Goal-Seeking Characters

189 **Example** **IQ Test** *WI*
Introducing Ogg, a person who will unlock and open a container when the player tells him to get something inside.

203 **Example** **The Man of Steel** *WI*
An escaping action which means "go to any room you can reach from here", and is only useful to non-player characters.

208 **Example** **The Man of Steel Excuses Himself** *WI*
Elaborating the report rules to be more interesting than "Clark goes west."

190 **Example** **Boston Cream** *WI*
A fuller implementation of Ogg, giving him a motivation of his own and allowing him to react to the situation created by the player.

§7.16. Social Groups

229 **Example** **Unthinkable Alliances** *WI*
People are to be grouped into alliances. To kiss someone is to join his or her faction, which may make a grand alliance; to strike them is to give notice of quitting, and to become a lone wolf.

231 **Example** **The Abolition of Love** *WI*
A thorough exploration of all the kinds of relations established so far, with the syntax to set and unset them.

436 **Example** **Lugubrious Pete's Delicatessen** *WI*
In this evocation of supermarket deli counter life, a list is used as a queue to keep track of who is waiting to be served.

176 **Example** **Strictly Ballroom** *WI*
People who select partners for dance lessons each turn.

353 **Example** **Emma** *WI*
Social dynamics in which groups of people form and circulate during a party.

Example **Happy Hour**

Listing visible characters as a group, then giving some followup details in the same paragraph about specific ones. 356 WI

Chapter 8: Vehicles, Animals and Furniture

§8.1. Bicycles, Cars and Boats

27 **Example** **Peugeot** WI
A journey from one room to another that requires the player to be on a vehicle.

104 **Example** **No Relation** WI
A car which must be turned on before it can be driven, and can only go to roads.

49 **Example** **Straw Boater** WI
Using text properties that apply only to some things and are not defined for others.

29 **Example** **Hover** WI
Letting the player see a modified room description when he's viewing the place from inside a vehicle.

§8.2. Ships, Trains and Elevators

9 **Example** **The Unbuttoned Elevator Affair** WI
A simple elevator connecting two floors which is operated simply by walking in and out, and has no buttons or fancy doors.

145 **Example** **Empire** WI
A train which follows a schedule, stopping at a number of different locations.

42 **Example** **Fore** WI
Understand "fore", "aft", "port", and "starboard", but only when the player is on a vessel.

271 **Example** **Dubai** WI
An elevator which connects any of 27 floors in a luxury hotel.

§8.3. Animals

406 **Example** **Feline Behavior** WI
A cat which reacts to whatever items it has handy, returning the result of a rulebook for further processing.

420 **Example** **Fido** *WI*
A dog the player can name and un-name at will.

99 **Example** **Today Tomorrow** *WI*
A few notes on "In the presence of" and how it interacts with concealed objects.

§8.4. Furniture

13 **Example** **Tamed** *WI*
Examples of a container and a supporter that can be entered, as well as nested rooms.

173 **Example** **Princess and the Pea** *WI*
The player is unable to sleep on a mattress (or stack of mattresses) because the bottom one has something uncomfortable under it.

361 **Example** **Kiwi** *WI*
Creating a raised supporter kind whose contents the player can't see or take from the ground.

432 **Example** **Circle of Misery** *WI*
Retrieving items from an airport luggage carousel is such fun, how can we resist simulating it, using a list as a ring buffer?

83 **Example** **Yolk of Gold** *WI*
Set of drawers where the item the player seeks is always in the last drawer he opens, regardless of the order of opening.

397 **Example** **Swigmore U.** *WI*
Adding a new kind of supporter called a perch, where everything dropped lands on the floor.

58 **Example** **U-Stor-It** *WI*
A "chest" kind which consists of a container which has a lid as a supporter.

§8.5. Kitchen and Bathroom

75 **Example** **Versailles** *WI*
A mirror which will reflect some random object in the room.

449 **Example** **Modern Conveniences** *WI*
Exemplifying the kind of source we might use in writing extensions for kitchen and bathroom appliances.

Chapter 9: Props: Food, Clothing, Money, Toys, Books, Electronics

§9.1. Food

48 **Example** **Would you...?** *WI*
Adding new properties to objects, and checking for their presence.

133 **Example** **Candy** *WI*
One of several identical candies chosen at the start of play to be poisonous.

143 **Example** **MRE** *WI*
Hunger that eventually kills the player, and foodstuffs that can delay the inevitable by different amounts of time.

391 **Example** **Stone** *WI*
A soup to which the player can add ingredients, which will have different effects when the player eats.

196 **Example** **Delicious, Delicious Rocks** *WI*
Adding a "sanity-check" stage to decide whether an action makes any sense, which occurs before any before rules, implicit taking, or check rules.

§9.2. Bags, Bottles, Boxes and Safes

298 **Example** **Safety** *WI*
A safe whose dial can be turned with SPIN SAFE TO 1131, and which will open only with the correct combination.

319 **Example** **Cinco** *WI*
A taco shell that can be referred to (when it contains things) in terms of its contents.

339 **Example** **Shipping Trunk** *WI*
A box of baking soda whose name changes to "completely ineffective baking soda" when it is in a container with something that smells funny.

344 **Example** **Unpeeled** *WI*
Calling an onion "a single yellow onion" when (and only when) it is being listed as the sole content of a room or container.

433 **Example** **Eyes, Fingers, Toes** *WI*
A safe with a multi-number combination, meant to be dialed over multiple turns, is implemented using a log of the last three numbers dialed. The log can then be compared to the safe's correct combination.

Example **Trachypachidae Maturin 1803** *WI*
340 Bottles with removable stoppers: when the stopper is in the bottle, the bottle is functionally closed, but the stopper can also be removed and used elsewhere. Descriptions of the bottle reflect its state intelligently.

Example **Hudsucker Industries** *WI*
342 Letters which are described differently as a group, depending on whether the player has read none, some, or all of them, and on whether they are alike or unlike.

Example **Fallout Enclosure** *WI*
35 Adding an enclosure kind that includes both containers and supporters in order to simplify text that would apply to both.

§9.3. Clothing

Example **Being Prepared** *WI*
56 A kind for jackets, which always includes a container called a pocket.

Example **Some Assembly Required** *WI*
332 Building different styles of shirt from component sleeves and collars.

Example **Hays Code** *WI*
338 Clark Gable in a pin-striped suit and a pink thong.

Example **What Not To Wear** *WI*
242 A general-purpose clothing system that handles a variety of different clothing items layered in different combinations over different areas of the body.

Example **Get Me to the Church on Time** *WI*
46 Using kinds of clothing to prevent the player from wearing several pairs of trousers at the same time.

Example **Bogart** *WI*
234 Clothing for the player that layers, so that items cannot be taken off in the wrong order, and the player's inventory lists only the clothing that is currently visible.

§9.4. Money

Example **Widget Enterprises** *WI*
263 Allowing the player to set a price for a widget on sale, then determining the resulting sales based on consumer demand, and the resulting profit and loss.

Example **Frozen Assets**

A treatment of money which keeps track of how much the player has on him, and a BUY command which lets him go shopping. 264 WI

Example **Money for Nothing** 265 WI
An OFFER price FOR command, allowing the player to bargain with a flexible seller.

Example **Fabrication** 269 WI
A system of assembling clothing from a pattern and materials; both the pattern and the different fabrics have associated prices.

Example **Nickel and Dimed** 262 WI
A more intricate system of money, this time keeping track of the individual denominations of coins and bills, specifying what gets spent at each transaction, and calculating appropriate change.

Example **Introduction to Juggling** 283 WI
Assortment of equipment defined with price and description, in a table.

§9.5. Dice and Playing Cards

Example **Do Pass Go** 129 WI
A pair of dice which can be rolled, and are described with their current total when not carried, and have individual scores when examined.

Example **Wonka's Revenge** 175 WI
A lottery drum which redistributes the tickets inside whenever the player spins it.

Example **Jokers Wild** 275 WI
A deck of cards which can be shuffled and dealt from.

Example **Tilt 1** 318 WI
A deck of cards with fully implemented individual cards, which can be separately drawn and discarded, and referred to by name.

Example **Tilt 2** 407 WI
A deck of cards with fully implemented individual cards; when the player has a full poker hand, the inventory listing describes the resulting hand accordingly.

§9.6. Reading Matter

Example **The Trouble with Printing** 292 WI
Making a READ command, distinct from EXAMINE, for legible objects.

303 **Example** **Pages** *WI*
A book with pages that can be read by number (as in "read page 3 in...") and which accepts relative page references as well (such as "read the last page of...", "read the next page", and so on).

443 **Example** **The Fourth Body** *WI*
Notebooks in which the player can record assorted notes throughout play.

324 **Example** **Bibliophilia** *WI*
A bookshelf with a number of books, where the player's command to examine something will be interpreted as an attempt to look up titles if the bookshelf is present, but otherwise given the usual response.

335 **Example** **AARP-Gnosis** *WI*
An Encyclopedia set which treats volumes in the same place as a single object, but can also be split up.

444 **Example** **The Fifth Body** *WI*
An expansion on the notebook, allowing the player somewhat more room in which to type his recorded remark.

94 **Example** **Costa Rican Ornithology** *WI*
A fully-implemented book, answering questions from a table of data, and responding to failed consultation with a custom message such as "You flip through the Guide to Central American Birds, but find no reference to penguins."

§9.7. Painting and Labeling Devices

308 **Example** **Palette** *WI*
An artist's workshop in which the canvas can be painted in any colour, and where painterly names for pigments ("cerulean") are accepted alongside everyday ones ("blue").

36 **Example** **Brown** *WI*
A red sticky label which can be attached to anything in the game, or removed again.

55 **Example** **Early Childhood** *WI*
A child's set of building blocks, which come in three different colours - red, green and blue - but which can be repainted during play.

§9.8. Simple Machines

61 **Example** **Control Center** *WI*
Objects which automatically include a description of their component parts whenever they are examined.

427 Example **What Makes You Tick** WI
Building a fishing pole from several component parts that the player might put together in any order.

57 Example **Model Shop** WI
An "on/off button" which controls whatever device it is part of.

53 Example **Signs and Portents** WI
Signpost that points to various destinations, depending on how the player has turned it.

§9.9. Televisions and Radios

66 Example **Radio Daze** WI
A radio that produces a cycle of output using varying text.

311 Example **Aspect** WI
Understanding aspect ratios (a unit) in the names of televisions.

313 Example **Channel 1** WI
Understanding channels (a number) in the names of televisions.

316 Example **Channel 2** WI
Understanding channels (a number) in the names of televisions, with more sophisticated parsing of the change channel action.

336 Example **Aftershock** WI
Modifying the rules for examining a device so that all devices have some specific behavior when switched on, which is described at various times.

§9.10. Telephones

226 Example **Four Cheeses** WI
A system of telephones on which the player can call distant persons and have conversations.

253 Example **Alias** WI
A telephone with phone numbers of the standard American seven-digit length.

§9.11. Clocks and Scientific Instruments

180 Example **Witnessed 2** WI
A piece of ghost-hunting equipment that responds depending on whether or not the meter

is on and a ghost is visible or touchable from the current location.

299 **Example** **Tom's Midnight Garden** *WI*
A clock kind that can be set to any time using "the time understood"; may be turned on and off; and will advance itself only when running. Time on the face is also reported differently depending on whether the clock is analog or digital.

365 **Example** **Ginger Beer** *WI*
A portable magic telescope which allows the player to view items in another room of his choice.

§9.12. Cameras and Recording Devices

273 **Example** **If It Hadn't Been For...** *WI*
A sound recording device that records the noises made by player and non-player actions, then plays them back on demand.

369 **Example** **Originals** *WI*
Allowing the player to create models of anything in the game world; parsing the name "model [thing]" or even just "[thing]" to refer to these newly-created models; asking "which do you mean, the model [thing] or the actual [thing]" when there is ambiguity.

417 **Example** **Mirror, Mirror** *WI*
The sorcerer's mirror can, when held up high, form an impression of its surroundings which it then preserves.

221 **Example** **Actor's Studio** *WI*
A video camera that records actions performed in its presence, and plays them back with time-stamps.

322 **Example** **Claims Adjustment** *WI*
An instant camera that spits out photographs of anything the player chooses to take a picture of.

Chapter 10: Physics: Substances, Ropes, Energy and Weight

§10.1. Gases

255 **Example** **Lethal Concentration 1** *WI*
A poisonous gas that spreads from room to room, incapacitating or killing the player when it reaches sufficient levels.

74 **Example** **Only You...** *WI*
Smoke which spreads through the rooms of the map, but only every other turn.

257 **Example** **Lethal Concentration 2** *WI*
Poisonous gas again, only this time it sinks.

§10.2. Liquids

117 **Example** **Thirst** *WI*
A waterskin that is depleted as the player drinks from it.

122 **Example** **Beverage Service** *WI*
A potion that the player can drink.

403 **Example** **Flotation** *WI*
Objects that can sink or float in a well, depending on their own properties and the state of the surrounding environment.

288 **Example** **Xylan** *WI*
Creating a new command that does require an object to be named; and some comments about the choice of vocabulary, in general.

192 **Example** **Frizz** *WI*
Liquid flows within containers and soaks objects that are not waterproof; any contact with a wet object can dampen our gloves.

193 **Example** **3 AM** *WI*
A shake command which agitates soda and makes items thump around in boxes.

266 **Example** **Lemonade** *WI*
Containers for liquid which keep track of how much liquid they are holding and of what kind, and allow quantities to be moved from one container to another.

267 **Example** **Savannah** *WI*
Using the liquid implementation demonstrated in Lemonade for putting out fires.

276 **Example** **Noisy Cricket** *WI*
Implementing liquids that can be mixed, and the components automatically recognized as matching one recipe or another.

333 **Example** **Lakeside Living** *WI*
Similar to "Lemonade", but with bodies of liquid that can never be depleted, and some adjustments to the "fill" command so that it will automatically attempt to fill from a large liquid source if possible.

§10.3. Dispensers and Supplies of Small Objects

378 **Example** **Pizza Prince** *WI*
Providing a pizza buffet from which the player can take as many pieces as he wants.

124 **Example** **Extra Supplies** *WI*
A supply of red pens from which the player can take another pen only if he doesn't already have one somewhere in the game world.

§10.4. Glass and Other Damage-Prone Substances

97 **Example** **Ming Vase** *WI*
ATTACK or DROP break and remove fragile items from play.

123 **Example** **Spring Cleaning** *WI*
A character who sulks over objects that the player has broken (and which are now off-stage).

195 **Example** **Paddington** *WI*
A CUT [something] WITH [something] command which acts differently on different types of objects.

314 **Example** **Terracottissima** *WI*
The flowerpots once again, but this time arranged so that after the first breakage all undamaged pots are said to be "unbroken", to distinguish them from the others.

404 **Example** **Kyoto** *WI*
Expanding the effects of the THROW something AT something command so that objects do make contact with one another.

§10.5. Volume, Height, Weight

232 **Example** **Swerve left? Swerve right? Or think about it and die?** *WI*
Building a marble chute track in which a dropped marble will automatically roll downhill.

268 **Example** **Depth** *WI*
Receptacles that calculate internal volume and the amount of room available, and cannot be overfilled.

258 **Example** **Dimensions** *WI*
This example draws together the previous snippets into a working implementation of the weighbridge.

270 **Example** **The Speed of Thought** *WI*
Describing scientifically-measured objects in units more familiar to the casual audience.

259 **Example** **Lead Cuts Paper** *WI*
To give every container a breaking strain, that is, a maximum weight of contents which it can bear - so that to put the lead pig into a paper bag invites disaster.

§10.6. Ropes

228 **Example** **Otranto** *WI*
A kind of rope which can be tied to objects and used to anchor the player or drag items from room to room.

261 **Example** **Snip** *WI*
A string which can be cut into arbitrary lengths, and then tied back together.

§10.7. Electricity and Magnetism

346 **Example** **Rules of Attraction** *WI*
A magnet which picks up nearby metal objects, and describes itself appropriately in room descriptions and inventory listings, but otherwise goes by its ordinary name.

408 **Example** **Electrified** *WI*
Adding a rule before the basic accessibility rule that will prevent the player from touching electrified objects under the wrong circumstances.

139 **Example** **Witnessed 1** *WI*
A kind of battery which can be put into different devices, and which will lose power after extended use.

§10.8. Fire

118 **Example** **Thirst 2** *WI*
A campfire added to the camp site, which can be lit using tinder.

150 **Example** **Bruneseau's Journey** *WI*
A candle which reacts to lighting and blowing actions differently depending on whether it has already been lit once.

418 **Example** **The Cow Exonerated** *WI*
Creating a class of matches that burn for a time and then go out, with elegant reporting when several matches go out at once.

401 **Example** **In Fire or in Flood** *WI*
A BURN command; flammable objects which light other items in their vicinity and can burn for different periods of time; the possibility of having parts or contents of a flaming item which survive being burnt.

§10.9. Heat

84 **Example** **Grilling** *WI*
A grill, from which the player is not allowed to take anything lest he burn himself.

182 **Example** **Entropy** *WI*
All objects in the game have a heat, but if not kept insulated they will tend toward room temperature (and at a somewhat exaggerated rate).

326 **Example** **Masochism Deli** *WI*
Multiple potatoes, with rules to make the player drop the hot potato first and pick it up last.

331 **Example** **Hot Glass Looks Like Cold Glass** *WI*
Responding to references to a property that the player isn't yet allowed to mention (or when not to use "understand as a mistake").

§10.10. Magic (Breaking the Laws of Physics)

215 **Example** **Magneto's Revenge** *WI*
Kitty Pryde of the X-Men is able to reach through solid objects, so we might implement her with special powers that the player does not have...

225 **Example** **Interrogation** *WI*
A wand which, when waved, reveals the concealed items carried by people the player can see.

227 **Example** **Transmutations** *WI*
A machine that turns objects into other, similar objects.

398 **Example** **Access All Areas** *WI*
The Pointy Hat of Liminal Transgression allows its wearer to walk clean through closed doors.

§10.11. Mathematics

240 **Example** **Number Study** *WI*
The parity and joint magnitude relations explored.

434 **Example** **The Fibonacci Sequence** *WI*
The modest Leonardo Fibonacci of Pisa will be only too happy to construct his sequence on request, using an array.

437 **Example** **Sieve of Eratosthenes** *WI*
The haughty Eratosthenes of Cyrene will nevertheless consent to sieve prime numbers on request.

Chapter 11: Out Of World Actions and Effects

§11.1. Start-Up Features

382 **Example** **Bikini Atoll** *WI*
Delaying the banner for later.

440 **Example** **Alien Invasion Part 23** *WI*
Keeping a preference file that could be loaded by any game in a series.

126 **Example** **Hatless** *WI*
It's tempting to use "now..." to distribute items randomly at the start of play, but we need to be a little cautious about how we do that.

§11.2. Saving and Undoing

210 **Example** **Spellbreaker** *WI*
P. David Lebling's classic "Spellbreaker" (1986) includes a room where the game cannot be saved: here is an Inform implementation.

211 **Example** **A point for never saving the game** *WI*
In some of the late 1970s "cave crawl" adventure games, an elaborate scoring system might still leave the player perplexed as to why an apparently perfect play-through resulted in a score which was still one point short of the supposed maximum. Why only 349 out of 350? The answer varied, but sometimes the last point was earned by never saving the game - in other words by playing it right through with nothing to guard against mistakes (except perhaps UNDO for the last command), and in one long session.

§11.3. Helping and Hinting

111 **Example** **Y ask Y?** *WI*
Noticing when the player seems to be at a loss, and recommending the use of hints.

284 **Example** **Food Network Interactive** *WI*
Using a menu system from an extension, but adding our own material to it for this game.

296 **Example** **Ish.** *WI*
A (very) simple HELP command, using tokens to accept and interpret the player's text whatever it might be.

329 **Example** **Query** *WI*
Catching all questions that begin with WHO, WHAT, WHERE, and similar question words, and responding with the instruction to use commands, instead.

285 **Example** **Trieste** *WI*
Table amendment to adjust HELP commands provided for the player.

400 **Example** **Solitude** *WI*
Novice mode that prefaces every prompt with a list of possible commands the player could try, and highlights every important word used, to alert players to interactive items in the scenery.

54 **Example** **Real Adventurers Need No Help** *WI*
Allowing the player to turn off all access to hints for the duration of a game, in order to avoid the temptation to rely on them overmuch.

230 **Example** **The Unexamined Life** *WI*
An adaptive hint system that tracks what the player needs to have seen or to possess in order to solve a given puzzle, and doles out suggestions accordingly. Handles changes in the game state with remarkable flexibility, and allows the player to decide how explicit a nudge he wants at any given moment.

§11.4. Scoring

219 **Example** **Bosch** *WI*
Creating a list of actions that will earn the player points, and using this both to change the score and to give FULL SCORE reports.

136 **Example** **Mutt's Adventure** *WI*
Awarding points for visiting a room for the first time.

137 **Example** **No Place Like Home** *WI*
Recording a whole table of scores for specific treasures.

166 **Example** **Panache** *WI*
Replacing the score with a plot summary that records the events of the plot, scene by scene.

Example **Goat-Cheese and Sage Chicken**

Implementing a FULL SCORE command which lists more information than the regular SCORE command, adding times and rankings, as an extension of the example given in this chapter. WI

Example **Rubies** WI
A scoreboard that keeps track of the ten highest-scoring players from one playthrough to the next, adding the player's name if he has done well enough.

§11.6. Ending The Story

Example **Battle of Ridgefield** WI
Completely replacing the endgame text and stopping the game without giving the player a chance to restart or restore.

Example **Finality** WI
Not mentioning UNDO in the final set of options.

Example **Jamaica 1688** WI
Adding a feature to the final question after victory, so that the player can choose to reveal notes about items in the game.

Example **Xerxes** WI
Offering the player a menu of things to read after winning the game.

Example **Labyrinth of Ghosts** WI
Remembering the fates of all previous explorers of the labyrinth.

Example **Big Sky Country** WI
Allowing the player to continue play after a fatal accident, but penalizing him by scattering his possessions around the game map.

Chapter 12: Typography, Layout, and Multimedia Effects

§12.1. Typography

Example **Garibaldi 2** WI
Adding coloured text to the example of door-status readouts.

Example **Chanel Version 1** WI
Making paired italic and boldface tags like those used by HTML for web pages.

Example **Tilt 3**

Displaying the card suits from our deck of cards with red and black colored unicode symbols. 450 WI

70 Example **The Über-complète clavier** WI
This example provides a fairly stringent test of exotic lettering.

§12.2. The Status Line

114 Example **Politics as Usual** WI
Have the status line indicate the current region of the map.

350 Example **Ways Out** WI
A status line that lists the available exits from the current location.

387 Example **Blankness** WI
Emptying the status line during the first screen of the game.

412 Example **Capital City** WI
To arrange that the location information normally given on the left-hand side of the status line appears in block capitals.

452 Example **Status line with centered text, the hard way** WI
A status line which has only the name of the location, centered.

351 Example **Guided Tour** WI
A status line that lists the available exits from the current location, changing the names of these exits depending on whether the room has been visited or not.

115 Example **Centered** WI
Replacing the two-part status line with one that centers only the room name at the top of the screen.

§12.3. Footnotes

300 Example **Ibid.** WI
A system which allows the author to assign footnotes to descriptions, and permits the player to retrieve them again by number, using "the number understood". Footnotes will automatically number themselves, according to the order in which the player discovers them.

§12.5. Glulx Multimedia Effects

445 **Example** **Flathead News Network** *WI*
Using external files, together with a simple Unix script running in the background, to provide live news headlines inside a story file.

Chapter 13: Testing and Publishing

§13.1. Testing

2 **Example** **Bic** *WI*
Testing to make sure that all objects have been given descriptions.

414 **Example** **Alpha** *WI*
Creating a beta-testing command that matches any line starting with punctuation.

§13.2. Publishing

446 **Example** **Baedeker** *WI*
Creating a floorplan of the cathedral using the locations from previous examples.

447 **Example** **Port Royal 5** *WI*
Port Royal scenario given instructions for an EPS map.

448 **Example** **Bay Leaves and Honey Wine** *WI*
Creating a map of Greece using the locations from previous examples.

General Index

Examples in Alphabetical Order
Examples in Thematic Order
Examples in Numerical Order
Part I. Writing with Inform
Part II. The Inform Recipe Book

#

`!`, ending sentences 2.3, 5.2

`"[']"` *say phrase* 5.2

`"`

defining texts 2.3, 3.1, 5.1, 5.2

not usable in unit notations 15.14

`"[bold type]"` *say phrase* *12.1*

`"[fixed letter spacing]"` *say phrase* *12.1*

`"[italic type]"` *say phrase* *12.1*

`"[roman type]"` *say phrase* *12.1*

`"[variable letter spacing]"` *say phrase* *12.1*

`'`, meaning `"` 2.3, 5.2, 25.2

(arithmetic value) `*` *(arithmetic value)* *phrase* 15.5

(arithmetic value) `+` *(arithmetic value)* *phrase* 15.5

`--`

optional words in phrase definitions 11.3

optional words in Understand grammar 17.12

--, as a blank table entry 16.7

--, separating cases 11.8

(arithmetic value) - *(arithmetic value)* *phrase* 15.5

->, in phrase kinds 22.3

.

before text substitutions 5.6

ending sentences with line breaks 2.3, 5.2, 5.8

(arithmetic value) / *(arithmetic value)* *phrase* 15.5

/

separating synonymous words in grammar 17.4, 17.12, 17.13

separating synonymous words in phrase definitions 11.3

separating synonymous words in topics 16.13

: 6.4, 11.2, 11.7, 11.16, 19.3

,

comma replacing colon 7.2

displaying serial comma 2.12, 5.5

in list constants, requiring spaces 21.2

separating phrase options 11.14

?, ending sentences 2.3, 5.2

[ ]

comments 2.3

text substitutions 2.3, 5.1, 5.2

(+ +), for including Inform 7 code in Inform 6 27.19

(- -), for including Inform 6 code in Inform 7 27.15, 27.19

( )

for calling values 8.15

for defining mistakes 17.21

for implicit multiplication 15.18

for phrase parameters 11.3

in parser clarifications 18.30

in response names 14.10

{ }

for constant lists 21.2

for including Inform 7 expressions in Inform 6 27.15

|, for paragraph break 2.3

The Über-complète clavier *example* 5.11 ex 70, *12.1 ex 70*

A

A&E *example* 7.13 ex 101, *3.2 ex 101*

a / an / the 3.18, 5.3, 5.5

AARP-Gnosis *example* 18.5 ex 335, *9.6 ex 335*

abandon the *(activity)* activity *phrase* 18.7

abandon the *(activity on values)* activity with *(value)* *phrase* 18.7

abbreviated room descriptions *use option* 2.12

abbreviations

chains of equality tests 11.8

object names 3.2

property names, omitting of the 8.6

rule actions with doing it 7.17

abide by *(rule)* *phrase* 19.14

abide by *(values based rule producing values)* for *(value)* *phrase* 19.14

able to see *relation verb* *see* visibility

able to touch *relation verb* *see* touchability

The Abolition of Love *example* 13.10 ex 231, *7.16 ex 231*

About Inform's regular expression support *example* 20.6 ex 415, *1.4 ex 415*

About the examples *example* 1.1 ex 1, *1.1 ex 1*

absolute value of *(real number)* *phrase* 15.6

ABSTRACT *testing command* 24.4

accented letters *see* characters (letters)

Access All Areas *example* 19.5 ex 398, *10.10 ex 398*

access, deny / allow access *(reaching inside / outside)* *outcome phrase* 12.18

accessibility text for figures and sounds 23.9

Actaeon *example* 17.7 ex 302, *7.13 ex 302*

action name 12.20

action name part of *(action)* *phrase* 12.20

`action of` *(an action)* *phrase* 12.20

`action processing` *rulebook* 19.15

actions 7.1, 12.1, 12.21

applying to multiple objects 17.4, *6.15*

as conditions 9.12

as values 12.20

`continue the action` *phrase* 7.3, 7.5

defining new actions 12.7, *6.1*

defining new actions in Inform 6 27.22

`exiting` 7.14

`going`, special cases 7.13, 7.14

guidelines for action rules 12.21

instructing other characters

persuasion rules 12.4

unsuccessful attempts 12.5

instructing other people 12.3, *7.14*

involving senses 7.7, *3.8*

involving speech 7.6

irregular verbs in action names 12.8

multiple action processing 17.20

new kinds of action 7.15

not the same as activities 18.1

`out of world` 12.15, 12.21

performing with `try` 7.4, 12.1, 12.6

processing sequence 12.2, 19.15, *6.3*

`pushing` objects to other rooms 7.14

reaching inside / outside rules 12.16, 12.18

reactions by other characters *7.3*

redirecting actions *6.14*

referring to current action with `action` 12.7

requirements for actions 12.16, 12.17, 12.18, 12.19, *6.2*

rules applying only when characters are present 7.12

rules applying to rooms and regions 7.11, 7.13

rules applying under arbitrary conditions, using `when` 7.12

rules for actions found in rulebooks 12.2, *6.3*

rules for all but some actions 7.9

rules for any actor (including the player) 12.14

rules for consecutive repetitions 7.17

rules for multiple actions 7.8

rules for new actions 12.9, *6.2*

rules for other characters 12.4, 12.5, 12.11, 12.12, 12.13

rules for repetitions 7.16

spontaneous by other people 12.6

`stop the action` *phrase* 7.3

stored actions 12.20

suppressing success messages with `try silently` 7.4, 12.1

understanding as mistakes 17.21, *5.3*

understanding in player's command 17.1

using after rules to change output 7.5

using before rules to test or change outcomes first 7.3

using instead rules to change outcome 7.2

variables for actions 12.10

ACTIONS *testing command* 7.1, 12.1, 12.5, 12.12, 24.3, *10.10*, *13.1*

Actions page of Index panel *user interface* 12.5, 12.21, 17.1, 17.2, 17.3, 24.3

`active / inactive` *(use option)* *adjective* 27.18

activities 18.1, 18.2

abandoning 18.7, 18.25

beginning/ending by hand 18.7

catalogue 18.8

`amusing a victorious player` 18.39

`asking which do you mean` 18.31

`choosing notable locale objects for something` 18.27

`clarifying the parser's choice of something` 18.30

`constructing the status line` 18.23

deciding the concealed possessions of something 3.24, 18.9

deciding the scope of something 18.29, *5.5*

deciding whether all includes 18.36, *6.15*

grouping together something 18.14

implicitly taking something 12.17, 18.34, *6.8*

issuing the response text of something 18.15

listing contents of something 18.13

listing nondescript items of something 18.25, *3.1*

printing a locale paragraph about 18.28, *3.1*

printing a number of something 18.12

printing a parser error 18.35

printing a refusal to act in the dark 18.18

printing inventory details of something 18.17

printing room description details of something 18.16, *3.1*

printing the announcement of darkness 18.19

printing the announcement of light 18.20

printing the banner text 18.37

printing the description of a dark room 18.22

printing the locale description of something 18.26

printing the name of a dark room 18.21

printing the name of something 18.10

printing the player's obituary 18.38

printing the plural name of something 18.11

reading a command 18.33

starting the virtual machine 18.40

supplying a missing noun/second noun 18.32

writing a paragraph about 18.24, *3.1*

checking for ongoing activities in rules 18.4

defining 18.5

empty / non-empty *adjective* 18.5

golden rules for constructing 18.2, 18.7

not the same as actions 18.1

rules for activities 18.3

variables for activities 18.6

Actor's Studio *example* 12.20 ex 221, *9.12 ex 221*

actor part of *(action)* *phrase* 12.20

actor — *person* *action variable* 12.3, 12.14, 14.4

actors *see* characters (people)

adaptive displayed text 14.2, *2.1*

contractions 14.7, 14.8

pronouns 14.2, 14.4, 14.5, 14.6, 14.8

verbs 14.2, 14.3

`add` *(list of values)* `at entry` *(number)* `in` *(list of values) phrase* 21.5

`add` *(list of values)* `to` *(list of values) phrase* 21.5

`add` *(value)* `at entry` *(number)* `in` *(list of values) phrase* 21.5

`add` *(value)* `to` *(list of values) phrase* 21.1, 21.5

adjacency *relation* 6.14, 13.8

`adjacent` *(room) adjective* 6.14

adjectives 6.3

- defining 6.4
  - with Inform 6 27.24
  - with `to decide whether` 11.16
- for numeric quantities 6.6, 6.7, 6.8
- for numeric quantities with units 15.12
- for values 6.5
- in descriptions 6.2

adverbs *6.18*

(time) `after` *(time) phrase* 9.9

(name of kind) `after` *(enumerated value) phrase* 11.18

`after` *(action) rulebook* 7.5

- guidelines for use 12.21
- in action processing sequence 12.2, *6.3*
- not used for `out of world` actions 12.15

`after` *(activity) rulebook* 18.3

`after`, ordering for rules 19.4

Aftershock *example* 18.5 ex 336, *9.9 ex 336*

Age of Steam *example* 10.3 ex 156, *4.4 ex 156*

Ahem *example* 11.3 ex 169, *2.1 ex 169*

Air Conditioning is Standard *example* 18.24 ex 354, *4.3 ex 354*

Albaugh, Lea 25.9

Alias *example* 15.2 ex 253, *9.10 ex 253*

Alien Invasion Part 23 *example* 23.13 ex 440, *11.1 ex 440*

`all except`, in descriptions 6.16

`all`, in descriptions 6.15

ALL, including things in 18.36, *6.15*

All Roads Lead to Mars *example* 6.14 ex 78, *3.2 ex 78*

`allow access` *(reaching inside / outside)* *outcome phrase* 12.18

`almost all`, in descriptions 6.15

Alpaca Farm *example* 17.2 ex 289, *6.17 ex 289*

Alpha *example* 20.6 ex 414, *13.1 ex 414*

`alter the multiple object list to` *(list of objects)* *phrase* 17.20

`always / never`, in assertions 4.3

`always`, in defining constant values 4.13

`amended`, for tables 16.19

`American dialect` *use option* 2.12, 2.13, 5.4

AMUSING 9.4, 18.39, *11.6*

amusing a victorious player *activity* 18.39

an / a / the 3.18, 5.3, 5.5

Anchorite *example* 17.2 ex 290, *6.10 ex 290*

animals *kind* 3.17, 4.1, *8.3*

rideable 3.16, *8.3*

announcement of darkness -- printing the announcement of darkness *activity* 18.19

announcement of light -- printing the announcement of light *activity* 18.20

Annoyotron Jr *example* 9.15 ex 153, *7.2 ex 153*

anonymously abide by *(rule)* *phrase* 19.14

anonymously abide by *(values based rule producing values)* for *(value)* *phrase* 19.14

another, in defining relations 13.6

answering *(actor)* *action* 7.6, 17.10

Ant-Sensitive Sunglasses *example* 18.1 ex 334, *3.1 ex 334*

Anteaters *example* 12.20 ex 222, *6.14 ex 222*

[any] *grammar token* 17.7, *5.5*

[anybody] *grammar token* 17.7

anybody, in descriptions 6.2

[anyone] *grammar token* 17.7

anyone, in descriptions 6.2

[anything] *grammar token* 17.7

`anything`, in descriptions 6.2

`[anywhere]` *grammar token* 17.7

`anywhere`, in descriptions 6.2

apostrophe *see* punctuation: apostrophe, meaning quotation mark

appearance *see* descriptions (displayed)

`append` *(text)* `to` *(external file)* *phrase* 23.14

Apples *example* 18.31 ex 368, *6.17 ex 368*

(phrase nothing -> value) `applied` *phrase* 22.3

(phrase (value, value, value) -> value) `applied to` *(value)* `and` *(value)* `and` *(value)* *phrase* 22.3

(phrase (value, value) -> value) `applied to` *(value)* `and` *(value)* *phrase* 22.3

(phrase value -> value) `applied to` *(value)* *phrase* 22.3

(phrase value -> value) `applied to` *(list of values)* *phrase* 22.5

`apply` *(phrase nothing -> nothing)* *phrase* 22.3

`apply` *(phrase value -> nothing)* `to` *(value)* *phrase* 22.3

`apply` *(phrase (value, value) -> nothing)* `to` *(value)* `and` *(value)* *phrase* 22.3

`apply` *(phrase (value, value, value) -> nothing)* `to` *(value)* `and` *(value)* `and` *(value)* *phrase* 22.3

`applying to`, in defining actions 12.7, 17.1

Approximate Metric Units *extension* 15.13

`arccosine of` *(real number)* *phrase* 15.7

Archive *web site* *see* IF Archive

`arcsine of` *(real number)* *phrase* 15.7

`arctangent of` *(real number)* *phrase* 15.7

`"[are]"` *say phrase* 14.2

`"[aren't]"` *say phrase* 14.8

arguments *see* parameters

arithmetic *see* calculation

`arithmetic value` 22.1, 22.6

The Art of Noise *example* 7.7 ex 95, *3.8 ex 95*

articles 3.18

displaying and capitalizing 7.10

`as something new` 17.3

`asking` *(actor)* `about` *action* 7.6

`asking` *(actor)* `to try` *action* 12.3

asking the player yes/no 11.5

`asking which do you mean` *activity* 18.31

Aspect *example* 17.15 ex 311, *9.9 ex 311*

assemblies 4.15, *5.1*, *9.8*

assertions 2.1, 3.1

plural 4.4

`at least` *relation verb* 15.5

`at least`, in descriptions 6.16

`at most` *relation verb* 15.5

`at most`, in descriptions 6.16

`at` *(time)*`...` *assertion* 9.11, 19.7, *4.3*

(rule) `at` *(time)* *phrase* 9.11

audio *see* sounds

author *bibliographic data* 2.5, 18.37, 25.2

of an extension 27.4

`authorial modesty` *use option* 27.6

B

Babel, Treaty of Babel 25.5

`back side of` *(object)* *phrase* 3.12

backdrops *kind* 3.9, *3.4*, *3.9*

changing the location of backdrops 8.8

Backus-Naur form for rules *example* 19.7 ex 399, *1.4 ex 399*

Bad Hair Day *example* 7.2 ex 85, *5.3 ex 85*

Baedeker *example* 25.23 ex 446, *13.2 ex 446*

Ballpark *example* 5.4 ex 60, *2.1 ex 60*

banner text 18.37, 25.4, *11.1*, *11.5*

Baritone, Bass *example* 17.11 ex 309, *5.2 ex 309*

Barter Barter *example* 12.12 ex 207, *7.4 ex 207*

Basic Screen Effects *extension* 27.7, *4.5*

basis of rulebook 19.9

Battle of Ridgefield *example* 18.38 ex 383, *11.6 ex 383*

Bay Leaves and Honey Wine *example* 25.23 ex 448, *13.2 ex 448*

Beachfront *example* 7.12 ex 98, *6.6 ex 98*

Bee Chambers *example* 8.11 ex 125, *3.2 ex 125*

Beekeeper's Apprentice *example* 5.8 ex 68, *6.5 ex 68*

(time) `before` *(time)* *phrase* 9.9

(name of kind) `before` *(enumerated value)* *phrase* 11.18

`before` *(action)* *rulebook* 7.3

guidelines for use 12.21

in action processing sequence 12.2, *6.3*

not used for `out of world` actions 12.15

`before` *(activity)* *rulebook* 18.3

`before`, ordering for rules 19.4

`begin`, in phrases 11.7, 11.9

`begin the` *(activity)* `activity` *phrase* 18.7

`begin the` *(activity on values)* `activity with` *(value)* *phrase* 18.7

beginning of scene *see* scenes: rules run at beginning

beginning of story *see* initial state of the world

Being Peter *example* 19.12 ex 405, *7.10 ex 405*

Being Prepared *example* 4.15 ex 56, *9.3 ex 56*

Belfry *example* 3.18 ex 31, *7.1 ex 31*

Beneath the Surface *example* 13.10 ex 233, *6.6 ex 233*

`best route from` *(object)* `to` *(object)* *phrase* 6.14

`best route from` *(object)* `to` *(object)* `through` *(description of objects)* *phrase* 6.14

The Best Till Last *example* 17.20 ex 327, *6.15 ex 327*

Beverage Service *example* 8.10 ex 122, *10.2 ex 122*

bibliographic data 2.5, 18.37, 25.2, 25.4, 25.15, 25.17, 26.3, *11.5*

Bibliophilia *example* 17.17 ex 324, *9.6 ex 324*

Bic *example* 3.1 ex 2, *13.1 ex 2*

The Big Sainsbury's *example* 18.34 ex 377, *4.1 ex 377*

Big Sky Country *example* 9.4 ex 138, *11.6 ex 138*

Bikini Atoll *example* 18.37 ex 382, *11.1 ex 382*

`binary`, when declaring data files 23.12

Blackout *example* 20.8 ex 419, *2.1 ex 419*

blank entries in tables *see* tables: blank entries

Blank, Marc 2.13

`blank out` *(a table entry)* *phrase* 16.10

`blank out the whole of` *(table)* *phrase* 16.10

`blank out the whole row` *phrase* 16.10

`blank out the whole` *(table column)* `column in` *(table)* *phrase* 16.10

Blankness *example* 18.40 ex 387, *12.2 ex 387*

blessed transcripts *see* transcripts: in the Skein

Blink *example* 27.30 ex 454, *2.1 ex 454*

blorb file 23.6, 23.10, 25.4, 25.7, 26.3

blurb *bibliographic data* *see* story description

body parts 4.15, *5.1*

Bogart *example* 13.10 ex 234, *9.3 ex 234*

`book` *of source text* 2.5

books (implementing) *9.6*

consulting 7.6, *9.6*

with pages *9.6*

Bosch *example* 12.20 ex 219, *11.4 ex 219*

Boston Cream *example* 12.6 ex 190, *7.15 ex 190*

Bowler Hats and Baby Geese *example* 10.4 ex 159, *4.2 ex 159*

boxed quotations 5.12

`brace notation -- "[`*(list of values)*`in brace notation]"` *say phrase* 21.3

braces *see* punctuation: curly braces

brackets *see* punctuation: brackets

`break` *phrase* 11.12

breakable things (implementing) *10.4*

breaks *see* paragraph breaks

Bribery *example* 19.3 ex 392, *7.4 ex 392*

`BRIEF room descriptions` *use option* 2.12, *6.4*, *11.5*

Brooks, Frederick P. 1.1

Brown *example* 3.23 ex 36, *9.7 ex 36*

Bruneseau's Journey *example* 9.13 ex 150, *10.8 ex 150*

bugs in Inform 2.15

Bumping into Walls *example* 7.13 ex 102, *6.9 ex 102*

`burning` *action* *10.8*

`buying` *action* *9.4*

C

Cactus Will Outlive Us All *example* 12.20 ex 220, *6.14 ex 220*

calculation *10.11*

- arithmetic 15.5
- arithmetic with units 15.19, 15.20
- average of a numeric property 15.17
- conversion during calculations 15.3
- decrementing 8.12, 11.18
- exponential functions 15.6
- given by equations 15.18
- incrementing 8.12, 11.18
- logarithms 15.6
- rounding to the nearest multiple 9.8, 15.5
- rounding to the nearest whole number 15.3, 15.6
- total of a numeric property 15.17
- trigonometric functions 15.7
- with time 9.9, 9.10

`called`

- in creating assemblies 4.16

in creating rooms 3.2

in creating things 3.5

in defining action variables 12.10

in defining activity variables 18.6

in defining adjectives 6.4

in defining relations 13.5, 13.6, 13.12

in defining rulebook variables 19.10

in describing things 8.15, 11.11

Camp Bethel *example* 5.7 ex 67, *7.2 ex 67*

`"[can't]"` *say phrase* 14.7

`"[can]"` *say phrase* 14.2, 14.7

`"[can see]"` *say phrase* 14.2

`"[can touch]"` *say phrase* 14.2

Candy *example* 8.19 ex 133, *9.1 ex 133*

Capital City *example* 20.4 ex 412, *12.2 ex 412*

Carnivale *example* 12.16 ex 212, *3.4 ex 212*

`carried` *(thing)* *adjective* 3.20, 8.13

Carroll, Lewis 5.12

`"[carry]"` *say phrase* 14.2

`carry out` *(action)* *rulebook* 12.2, 12.9, *6.3*

guidelines for use 12.21

`carry out the` *(activity)* `activity` *phrase* 18.5

`carry out the` *(activity on values)* `activity with` *(value)* *phrase* 18.5

carrying *relation* 3.20, 8.7, 8.13, 13.4

requirement in defining actions 12.7, 12.17

`carrying capacity of` *(container/supporter/person)* *property* 3.19, *10.5*

avoid overflowing player's capacity using a hold-all 3.21

case sensitivity

checking texts 20.4

in matching texts 20.5

in real number constants 15.2

in regular expressions 20.6

in text substitutions 5.13

in text substitutions with lists 5.5

in text substitutions with objects 5.3, 7.10

in text substitutions with pronouns 14.4, 14.5, 14.6

Casino Banale *example* 18.27 ex 360, *5.5 ex 360*

Cater, John 23.10

Cave-troll *example* 20.8 ex 424, *6.17 ex 424*

`ceiling of` *(real number)* *phrase* 15.6

Celadon *example* 13.4 ex 224, *6.8 ex 224*

Centered *example* 8.3 ex 115, *12.2 ex 115*

chance *see* randomness

Chanel Version 1 *example* 27.28 ex 453, *12.1 ex 453*

change *see* now (a condition)

`change` *(direction)* `exit of` *(room)* `to nothing/nowhere` *phrase* 8.5

`change` *(direction)* `exit of` *(room)* `to` *(room)* *phrase* 8.5

`change` *(list of values)* `to have` *(number)* `entries/entry` *phrase* 21.10

Change of Basis *example* 4.7 ex 47, *6.11 ex 47*

`change the text of the player's command to` *(text)* *phrase* 18.33

Channel 1 *example* 17.15 ex 313, *9.9 ex 313*

Channel 2 *example* 17.15 ex 316, *9.9 ex 316*

`chapter` *of source text* 2.5

`character number` *(number)* `in` *(text)* *phrase* 20.3

characters (letters) 5.11

- capital letters as kind variables 22.7
- counting 20.3
- in text values 20.1
- replacing in text 20.8
- special / accented 5.10
- special meanings in regular expressions 20.6, 20.9
- Unicode (arbitrary symbols) 5.11, *12.1*
  - defining new names for 27.25

characters (people)

accepting gifts *7.4*, *9.4*

acting spontaneously 12.6, *7.2*

characterization *7.10*

combat *7.5*

dialogue *see* dialogue

giving instructions to other characters 12.3, *7.14*

goal-seeking *7.15*

groups of people *7.16*

Inanimate Listeners *extension* *7.7*

memory *see* knowledge (in story world)

moving around *7.13*

names and titles *7.1*

persuading 12.4

reacting to each other *7.16*

reacting to the player's actions *7.3*

reporting unsuccessful actions 12.5

rules applying only when characters are present 7.12

scripted conversations *7.12*

`check` *(action)* *rulebook* 12.2, 12.9, *6.3*

for other characters 12.12

guidelines for use 12.21

Cheese-makers *example* 10.9 ex 168, *7.8 ex 168*

Chimento, Philip 1.2

`choose a/the/-- blank row in/from` *(table name)* *phrase* 16.10

`choose a/the/-- random row in/from` *(table name)* *phrase* 16.5

`choose a/the/-- row` *(number)* `in/from` *(table name)* *phrase* 16.5

`choose a/the/-- row with` *(table column)* `of` *(value)* `in/from` *(table name)* *phrase* 16.5

`choosing notable locale objects for something` *activity* 18.27

Chronic Hinting Syndrome *example* 18.10 ex 341, *7.11 ex 341*

Cinco *example* 17.16 ex 319, *9.2 ex 319*

Circle of Misery *example* 21.11 ex 432, *8.4 ex 432*

Claims Adjustment *example* 17.16 ex 322, *9.12 ex 322*

`clarifying the parser's choice of something` *activity* 18.30

Cloak of Darkness *example* 17.2 ex 291, *3.7 ex 291*

clocks (implementing) *9.11*

`closed / open` *(container/door)* *property* 3.6, 3.13, 3.15, *9.2*

clothing *see* wearing

Cloves *example* 18.33 ex 373, *6.18 ex 373*

Clueless *example* 9.1 ex 135, *7.1 ex 135*

colon *see* punctuation: colon

columns, in tables *see* tables

combat *7.5*

comma *see* punctuation: comma

`command prompt` — *text global variable* 8.2, 9.7

commands (of the player) *see* understanding

comments

in source text 2.3

in transcripts from beta testers 17.21, 24.7, *13.1*

comparisons

counting 6.16

equality 11.6; *see also* equality

of enumerated values 11.18

of numbers 15.5

of numbers using adjectives 6.7

of numbers with units using adjectives 15.12

of text 20.5

with regular expressions 20.6

compass directions *see* directions

Complimentary Peanuts *example* 18.33 ex 376, *7.8 ex 376*

components 3.23

breaking apart *10.4*

for kinds 4.15

incorporation relation *see* incorporation

of bodies *5.1*

of machines *9.8*

concealed possessions 3.24, 18.9

`concealed` *(thing)* *adjective* 3.24, 18.9

concealment *relation* 3.24, 13.4, 18.9

in a room *see* hiding things from room descriptions by making them scenery

under other things *see* hiding things under other things

`condition`, naming properties 4.10

conditions

changing conditions of rules 19.5

defined by phrases 11.5

defining with `to decide whether` 11.16

equality 11.6

expressed as relations 13.12

for activity rules 18.4

for rules 7.12, 9.5, 19.5, 19.7

for scenes 10.4

for `understand` 17.17, *2.2*

historical 9.13

number of times 9.14

number of turns 9.15

warning concerning variables and other varying qualities 9.13, 9.15

testing 11.6

with `otherwise/else` 11.8

connections between rooms 3.2, *3.2*

changing 8.5

doors 3.12, *3.5*

inconsistent directions 3.3

one-way 3.3

rules applying to map connections 7.13

to nowhere 3.3

`consents -- if player consents` *phrase* 11.5

constant values 4.13

`constructing the status line` *activity* 18.23

`consulting` *(thing)* `about` *action* 7.6

`container in question` — *object global variable* 12.18

containers *kind* 3.5, *8.4*, *9.2*

carrying capacity 3.19, *10.5*

containment relation *see* containment

enterable 3.6

liquids in containers *10.2*

locked 3.13

open / closed 3.6, 3.15

pockets in clothing *9.3*

transparent / opaque 3.6, 3.15

vehicle as kind of container 3.16

containment *relation* 6.11, 8.13, 13.4

hierarchy, low-level interface 8.17

indirect 3.25, 6.11

placing the contents of something in scope 18.29

Contents page of Index panel *user interface* 2.6, 23.5, 27.10

`continue the action` *phrase* 7.3, 7.5, 19.11

`continue the activity` *phrase* 18.5

`continued`, for tables 16.18

contractions, adaptive contractions in displayed text 14.7, 14.8

Control Center *example* 5.5 ex 61, *9.8 ex 61*

conversation *see* dialogue

Copper River *example* 18.28 ex 362, *3.1 ex 362*

Cornelson, David 1.2

`corresponding to`, in tables 16.3

`cosine of` *(real number)* *phrase* 15.7

Costa Rican Ornithology *example* 7.6 ex 94, *9.6 ex 94*

`"[could]"` *say phrase* 14.7

`"[couldn't]"` *say phrase* 14.7

counted loops 11.10

counting 6.16

actions with `for the Nth time` 7.16, *11.4*

actions with `for the Nth turn` 7.17

characters / words / paragraphs / lines 20.3

conditions with `for the Nth time` 9.14

conditions with `for the Nth turn` 9.15

moves along a route 6.14

steps in an indirect relation 13.11

things 8.16

cover art 25.8

The Cow Exonerated *example* 20.7 ex 418, *10.8 ex 418*

The Crane's Leg 1 *example* 4.9 ex 52, *5.6 ex 52*

The Crane's Leg 2 *example* 19.3 ex 390, *5.6 ex 390*

creating *see* defining

Croft *example* 12.10 ex 202, *6.8 ex 202*

Crowther, Will 3.1

Crusoe *example* 18.5 ex 337, *6.5 ex 337*

`cube root of` *(arithmetic value)* *phrase* 15.5

Curare *example* 22.2 ex 439, *2.1 ex 439*

curly braces *see* punctuation: curly braces

`current action` *phrase* 12.20

cut *(snippet)* *phrase* 18.33

D

Daly, Liza 1.2

damaging things (implementing) *10.4*

dark / lighted *(room)* *property* 3.15, *3.7*

The Dark Ages Revisited *example* 12.9 ex 194, *3.7 ex 194*

darkness 3.15, 6.13, *3.7*

- printing a refusal to act in the dark *activity* 18.18
- printing the announcement of darkness *activity* 18.19
- printing the announcement of light *activity* 18.20
- printing the description of a dark room *activity* 18.22
- printing the name of a dark room *activity* 18.21
- rules to change perception of 12.19
- testing whether in darkness 11.5

data files *see* files (data files)

A Day For Fresh Sushi *example* 7.16 ex 112, *7.3 ex 112*

Day One *example* 10.4 ex 160, *4.2 ex 160*

Dearth and the Maiden *example* 7.15 ex 109, *5.3 ex 109*

death 9.4, 18.38, 18.39, *11.6*

debug version (of the story) *see* test version (of the story); test version (of the story)

debugging commands *see* testing commands

decide no *phrase* 11.16

decide on *(value)* *phrase* 11.17

`decide yes` *phrase* 11.16

`deciding the concealed possessions of something` *activity* 3.24, 18.9

`deciding the scope of something` *activity* 18.29, *5.5*

`deciding whether all includes` *activity* 18.36, *6.15*

decimals *see* real numbers

`say "[(real number) in decimal notation]"` *phrase* 15.4

`say "[(real number) to (number) decimal places]"` *phrase* 15.4

`decrease` *(a stored value)* `by` *(value)* *phrase* 8.12

`decrement` *(a stored value)* *phrase* 8.12

default outcome of a rulebook 19.11, 19.12

default properties of kinds 4.3; *see also* kinds: default values of kinds

`default value of` *(name of kind)* *phrase* 4.11

defining

- action variables 12.10
- actions 12.7, *6.1*
- activities 18.5
- activity variables 18.6
- adaptive displayable verbs 14.3
- adjectives 6.4
  - for numeric quantities 6.6, 6.7, 6.8
  - for numeric quantities with units 15.12

for values 6.5

with `to decide whether` 11.16

constant values 4.13

data files 23.12

directions 3.26

either/or properties 4.7

equations for values with units 15.18

figures 23.5

global variables 4.12

grammar tokens 17.13

producing values 17.14

Inform 6 entities *see* Inform 6 inclusions

input grammar 17.1

kinds 4.1

multiply using tables 16.16

kinds of thing 4.1, 4.2, 4.4

kinds of value 4.5, 4.9, 4.10

defining with a table 16.17

lists 21.2

by description 21.6, 21.7

phrases 11.2

plurals 4.4, *2.2*

prepositions for relations 13.10

previewing 23.5

relations 13.5, 13.13

relations expressing conditions 13.12

rooms 3.1

rulebook variables 19.10

rulebooks 19.8

rules 2.2, 19.3, 19.7

sentence verbs for properties 15.12

sentence verbs for relations 13.9

sounds 23.8

tables 16.1

temporary variables 11.15

testing commands 2.9, *13.1*

text substitutions 5.13

things 3.1

- multiply using tables 16.16
- multiply with counts 4.14
- using plurals 4.4
- using relations 4.15, 4.16

units of measure 15.8, 15.9, 15.10

units of measure with parts 15.14, 15.15

value properties 4.8, 4.9, 4.10

definite articles

when creating things 3.18

when displaying lists 5.5, 21.3

when displaying things 5.3

`definition` *assertion* 6.4, 19.7

for numeric quantities 6.6, 6.7, 6.8

for values 6.5

with phrase logic 11.16

(real number) `degrees` *phrase* 15.7

Delayed Gratification *example* 19.2 ex 389, *6.11 ex 389*

deleting things from the world 8.10

Delicious, Delicious Rocks *example* 12.9 ex 196, *9.1 ex 196*

Democratic Process *example* 7.3 ex 86, *6.8 ex 86*

`deny access` *(reaching inside / outside)* *outcome phrase* 12.18

Depth *example* 15.20 ex 268, *10.5 ex 268*

`described / undescribed` *(thing)* *property* 3.24

`describing`, in `understand as` 17.15

`description of a dark room -- printing the description of a dark room` *activity* 18.22

`description of` *(room)* *property* 3.10, 3.11

description of *(scene)* *property* 10.2

description of *(thing)* *property* 3.11

descriptions (displayed)

activities

choosing notable locale objects for something 18.27

grouping together something 18.14

listing contents of something 18.13

listing nondescript items of something 18.25, *3.1*

printing a locale paragraph about something 18.28, *3.1*

printing a number of something 18.12

printing inventory details of something 18.17

printing room description details of something 18.16, *3.1*

printing the description of a dark room 18.22

printing the locale description of something 18.26

printing the name of something 18.10

printing the plural name of something 18.11

writing a paragraph about 18.24, *3.1*

default descriptions using new kinds 4.2

full-length room descriptions / abbreviated room descriptions 2.12

hiding things from room descriptions 3.8

initial appearance of thing 3.11, 8.5, *3.1*

miscellaneous things in room 18.25, *3.1*

notable things in room 18.24, *3.1*

room contents 18.26, 18.27, 18.28, *3.1*, *6.4*

room descriptions with substituted text 5.1

scene descriptions 10.2

varying with properties 3.14

descriptions (references to things) 6.1

adjectives for magnitude 6.6

adjectives for values 6.5

as values 22.2

comparative adjectives 6.7

complex phrases 6.9

counting 6.16

defining adjectives for descriptions 6.4

determiners (all / none) 6.15

existence 6.10

listed in table 16.12

non-existence 6.10, 6.12

relative 6.9

superlative adjectives 6.8

syntax 6.2

determiners (all / none) 6.15

devices *kind* 3.14, *9.8*, *10.7*

displaying the on/off state *6.5*

dialogue *7.6*

ASK / TELL 7.6, *7.7*, *7.8*

keyword-based dialogue *7.8*

menu-based dialogue *7.8*

scripted conversations *7.12*

simple terms *7.7*

structure of conversation *7.9*

with inanimate objects *7.7*

`DICT_WORD_SIZE` *use option* *2.2*

Dimensions *example* 15.12 ex 258, *10.5 ex 258*

Dinner is Served *example* 12.18 ex 217, *3.6 ex 217*

`direction of` *(door)* `from` *(room)* *phrase* 3.12

directions *kind* 3.2, 3.5, 3.26, 4.1, *3.2*, *6.9*

defining 3.26

finding adjacent rooms 6.14

shipboard directions, FORE/AFT/PORT/STARBOARD *8.2*

disambiguation

of player commands 17.19, *6.17*

displaying parser's chosen thing 18.30

printing disambiguation request 18.31

of source text 2.6

precedence of phrase definitions 22.9, 22.10

precedence of rules 19.2, 19.4, 19.6, 19.16

Disenchantment Bay 1 *example* 3.7 ex 14, *1.3 ex 14*

Disenchantment Bay 2 *example* 3.8 ex 15, *1.3 ex 15*

Disenchantment Bay 3 *example* 3.9 ex 17, *1.3 ex 17*

Disenchantment Bay 4 *example* 3.11 ex 18, *1.3 ex 18*

Disenchantment Bay 5 *example* 3.12 ex 20, *1.3 ex 20*

Disenchantment Bay 6 *example* 3.13 ex 23, *1.3 ex 23*

Disenchantment Bay 7 *example* 3.14 ex 25, *1.3 ex 25*

Disenchantment Bay 8 *example* 3.16 ex 28, *1.3 ex 28*

Disenchantment Bay 9 *example* 3.17 ex 30, *1.3 ex 30*

Disenchantment Bay 10 *example* 3.20 ex 33, *1.3 ex 33*

Disenchantment Bay 11 *example* 3.21 ex 34, *1.3 ex 34*

Disenchantment Bay 12 *example* 3.23 ex 37, *1.3 ex 37*

`display` *(figure name)* *phrase* 23.6

`display the boxed quotation` *(text)* *phrase* 5.12

displaying descriptions *see* descriptions (displayed)

distance

measuring with `number of moves` 6.14

measuring with `number of steps` 13.11

(arithmetic value) `divided by` *(arithmetic value)* *phrase* 15.5

`do nothing` *phrase* 11.1

Do Pass Go *example* 8.18 ex 129, *9.5 ex 129*

`does the player mean` *rulebook* 17.19

`doing it`, a repeated action 7.17

`doing something / anything`, describing actions 7.9

`"[don't]"` *say phrase* 14.8

Don Pedro's Revenge *example* 8.2 ex 113, *7.5 ex 113*

`door` *(direction)* `from/of` *(room)* *phrase* 6.14

doors *kind* 3.12, *3.5*

- locked 3.13
- rules for going through doors 7.14

double-dash *see* punctuation: double-dash

Down Below *example* 3.14 ex 26, *3.7 ex 26*

Down in Oodville *example* 17.9 ex 304, *6.18 ex 304*

`drinking` *action* *10.2*

`dropping` *action* *6.8*

Dubai *example* 16.3 ex 271, *8.2 ex 271*

duplicates 4.14

- taken from a dispenser *10.3*

`during` *(scene)* 10.4, 19.7

dynamic memory allocation use option 20.2

E

e — real number constant 15.2

each, in descriptions 6.15

each other, in defining relations 13.6, 13.7

Early Childhood example 4.14 ex 55, *9.7 ex 55*

eating action 3.22, *9.1*

Eddystone example 12.17 ex 213, *3.4 ex 213*

edible / inedible *(thing)* property 3.22, *9.1*

either/or properties 3.6, 4.6

- changing 8.4
- defining 4.7
- used as adjectives 6.3

electricity (implementing) *10.7*

Electrified example 19.15 ex 408, *10.7 ex 408*

else *(a phrase)* phrase 11.8

else if *(a condition)* phrase 11.8

else unless *(a condition)* phrase 11.8

Elsie example 9.13 ex 151, *3.5 ex 151*

Emma example 18.24 ex 353, *7.16 ex 353*

Empire example 9.11 ex 145, *8.2 ex 145*

empty / non-empty adjective

- for activities 18.5

for lists 21.1, 21.2

for relations 13.14

for rulebooks 19.8

for tables 16.9

for texts 6.5, 20.3

enclosure *relation* 3.25

`end if` 11.7

`end`, in phrases 11.7, 11.9

end of scene *see* scenes: rules run at end

end of story 9.4, 18.38, 18.39, *11.6*

`end the` *(activity)* `activity` *phrase* 18.7

`end the` *(activity on values)* `activity with` *(value)* *phrase* 18.7

`end the story` *phrase* 9.4, *11.6*

`end the story finally` *phrase* 9.4, *11.6*

`end the story finally saying` *(text)* *phrase* 9.4, *11.6*

`end the story saying` *(text)* *phrase* 9.4, *11.6*

`end while` 11.9

`ended -- if` *(scene)* `has ended` *phrase* 10.4

Endurance *example* 19.15 ex 410, *4.1 ex 410*

`engineering notation` *use option* 15.2

English

defining plural forms 4.4, *2.2*

displaying numbers in words 5.4

displaying simple plurals 5.4

displaying times in words 9.7

forms for sentence verbs 13.9

forms of `to be` 13.10

irregular verbs 12.8, 13.9

narrative viewpoint, of standard responses 14.1, *5.6*

really understanding 2.16

simple superlative forms 6.8

tense

of conditions in rules 9.12, 9.13

of standard responses 14.1, *5.6*

`enterable` *(container) property* 3.6, *9.2*

vehicles made automatically enterable 3.16

`entering` *action* *6.10*

`Entire Game` — *scene constant* 10.3

Entrapment *example* 10.2 ex 155, *4.2 ex 155*

Entrevaux *example* 10.6 ex 163, *4.4 ex 163*

entries, in tables *see* tables

Entropy *example* 11.18 ex 182, *10.9 ex 182*

`entry` *(number)* `of` *(list)* 21.9

`enumerated value` 22.1, 22.6; *see also* kinds: of value

Epistemology *extension* *5.5*

`EPS file` *see* index map: customizing for release

equality *relation* 13.8

equations, for values with units 15.18

Equipment List *example* 11.14 ex 177, *6.7 ex 177*

`equivalence` *(relation)* *adjective* 13.14

`equivalent to` *(unit)* 15.10

error messages

- abject failures of Inform 2.15
- for player commands 18.35, *6.17*
- for source text 2.4

Escape *example* 3.12 ex 21, *3.6 ex 21*

Escape from the Seraglio *example* 19.15 ex 411, *6.15 ex 411*

Euler, Leonhard 15.2

Eve, Eric 1.2, *5.5*

`even / odd` *(number)* *adjective* 4.6, 6.5

events in the future 9.11

`every`

- creating assemblies 4.15, 4.16, *5.1*
- in descriptions 6.15

every turn *rulebook* 9.5, *7.2*

not for out of world actions 12.15

everywhere, placing backdrops 3.9, 8.8

exactly

counting conditions 9.14, 9.15

in descriptions 6.16

examining *action* *6.5*

handling READ separately *9.6*

except

describing actions 7.9

in descriptions 6.16

exclamation mark *see* punctuation: exclamation mark, ending sentences

existence

descriptions of any such thing 6.2

testing that such a thing exists 6.10

existent / nonexistent *(real number)* *adjective* 15.3

exiting *action* 7.14, *6.9*, *6.10*

exits *see* connections between rooms

exponential of *(real number)* *phrase* 15.6

extend *(list of values)* to *(number)* entries/entry *phrase* 21.10

extensions

built-in extensions 27.3

Inform 6 code in extensions 27.14

Inform 6 template layer 27.26, *6.3*

installing 2.10, 27.3

licensing of use 27.1, *1.1*

listing credits for 27.6, *11.5*

project-specific extensions 27.3

specific extensions

- Approximate Metric Units 15.13
- Basic Screen Effects 27.7, *4.5*
- Epistemology *5.5*
- Inanimate Listeners *7.7*, *9.6*
- Locksmith 2.10, 2.11, *3.5*, *9.2*
- Metric Units 15.13, 15.18, 15.20
- Mistype *6.17*
- Object Response Tests 24.6
- Punctuation Removal *6.13*, *7.1*, *7.7*
- Real Time Delays *12.4*
- Rideable Vehicles 3.16, *8.3*
- Skeleton Keys *9.2*
- Standard Rules 27.2, 27.14
- Unicode Character Names 5.11, 27.25

Unicode Full Character Names 5.11, 27.25

Vorple system 25.12

using 2.11

using specific versions 27.6

writing

author *bibliographic data* 27.4

documenting 27.11, 27.12

example 27.5

for a specific virtual machine 27.7

giving examples 27.12

including other extensions 27.8, 27.9

modifying other extensions 27.9

version number 27.6

external files *see* files (data files)

Extra Supplies *example* 8.10 ex 124, *10.3 ex 124*

The Eye of the Idol *example* 18.25 ex 357, *3.1 ex 357*

Eyes, Fingers, Toes *example* 21.11 ex 433, *9.2 ex 433*

F

Fabrication *example* 15.20 ex 269, *9.4 ex 269*

The Facts Were These *example* 21.10 ex 431, *6.15 ex 431*

failure

for actions 7.1, 12.2

for persuasion 12.4

for rules 19.11

Fallout Enclosure *example* 3.23 ex 35, *9.2 ex 35*

`false / true` 11.5

Farewell *example* 16.15 ex 281, *7.8 ex 281*

`fast route-finding` *use option* 6.14

for particular relations 13.11

Fate Steps In *example* 12.13 ex 209, *4.6 ex 209*

feelies 25.7

Feline Behavior *example* 19.13 ex 406, *8.3 ex 406*

`female / male / neuter` *(person)* *property* 3.17

Ferragamo Again *example* 11.3 ex 170, *7.10 ex 170*

The Fibonacci Sequence *example* 21.11 ex 434, *10.11 ex 434*

Fido *example* 20.8 ex 420, *8.3 ex 420*

The Fifth Body *example* 23.14 ex 444, *9.6 ex 444*

Fifty Times Fifty Ways *example* 5.13 ex 72, *2.1 ex 72*

Fifty Ways to Leave Your Larva *example* 5.13 ex 71, *2.1 ex 71*

`figure name` 23.10

figures 23.2, *12.5*

accessibility text 23.9

as values 23.10

cover art 25.8

declaring 23.5

displaying 23.6

formats allowed 23.4

placing in materials folder 23.4

files (compiling)

blorb file *see* blorb file

extensions 2.10, 27.3

materials folder *see* materials folder

Options.txt 2.13

files (data files) 23.11, *12.5*

declaring 23.12

exchanging with other programs 23.12, 23.15

loading initial state from a data file *11.1*

reading / writing / appending text 23.14

reading / writing tables 23.13

storing records of death in a data file *11.6*

`filter to` *(description of values)* `of` *(list of values)* *phrase* 22.5

`Final Question Options -- Table of Final Question Options` *11.6*

Finality *example* 18.38 ex 384, *11.6 ex 384*

`finally`, ending the story 9.4, 18.39, *11.6*

FIND *5.4*

Fine Laid *example* 7.4 ex 88, *6.14 ex 88*

Finishing School *example* 6.4 ex 73, *5.3 ex 73*

`finite / infinite` *(real number)* *adjective* 15.3

fire (implementing) *10.8*

`first`, listed first in rulebook 19.4, 19.7

First Name Basis *example* 3.5 ex 11, *2.2 ex 11*

`first person singular / plural` 14.1

`first thing held by` *(object)* *phrase* 8.17

`first time`, counting actions 7.16

`first time`, counting conditions 9.14

`first value of` *(name of kind)* *phrase* 11.18

Fisher, David *1.2*

`fixed in place / portable` *(thing)* *property* 3.6, 3.7, 8.7

Flashlight *example* 12.19 ex 218, *6.6 ex 218*

Flathead News Network *example* 23.15 ex 445, *12.5 ex 445*

Fleming, Ian 6.10

floating-point *see* real numbers

`floor of` *(real number)* *phrase* 15.6

Flotation *example* 19.9 ex 403, *10.2 ex 403*

fluids *see* liquids (implementing)

`follow` *(rule)* *phrase* 19.8

`follow` *(values based rule producing values)* `for` *(value)* *phrase* 19.9

fonts

fixed-width / variable-width 5.9, *12.1*

italic / bold / roman 5.9, *12.1*

food 3.22, *9.1*

Food Network Interactive *example* 16.18 ex 284, *11.3 ex 284*

footnotes (implementing) *12.3*

`for` *(activity)* *rulebook* 18.3

For Demonstration Purposes *example* 13.13 ex 239, *7.14 ex 239*

`for release only` 2.9, 25.6, *13.1*

Fore *example* 3.26 ex 42, *8.2 ex 42*

Formal syntax of sentences *example* 13.2 ex 223, *1.4 ex 223*

Formicidae *example* 21.6 ex 428, *6.15 ex 428*

Four Cheeses *example* 13.6 ex 226, *9.10 ex 226*

Four Stars 1 *example* 18.22 ex 349, *3.7 ex 349*

Four Stars 2 *example* 18.29 ex 364, *3.8 ex 364*

The Fourth Body *example* 23.14 ex 443, *9.6 ex 443*

fragile things (implementing) *10.4*

Fragment of a Greek Tragedy *example* 18.33 ex 374, *6.18 ex 374*

Frizz *example* 12.7 ex 192, *10.2 ex 192*

`from now --` *(rule)* `in` *(number)* `turn/turns from now` *phrase* 9.11

`from now --` *(rule)* `in` *(time)* `from now` *phrase* 9.11

`from -- inside / outside from` 3.2

`front side of` *(object)* *phrase* 3.12

Frozen Assets *example* 15.19 ex 264, *9.4 ex 264*

`full-length room descriptions` *use option* 2.12

Full Moon *example* 10.4 ex 157, *3.9 ex 157*

full stop *see* punctuation: full stop

Fun with Participles *example* 14.3 ex 245, *2.1 ex 245*

Further Reasons Why All Poets Are Liars *example* 12.10 ex 199, *3.3 ex 199*

`future / present / past tense` 14.1

future events 9.11

G

Galactic Patrol 20.6

game structure *see* story structure

games, dice and cards (implementing) *9.5*

Garibaldi 1 *example* 3.12 ex 22, *3.5 ex 22*

Garibaldi 2 *example* 5.9 ex 69, *12.1 ex 69*

gases (implementing) *10.1*

Generation X *example* 12.5 ex 188, *7.14 ex 188*

genre *bibliographic data* 25.2, 25.3

Get Axe *example* 12.12 ex 206, *6.10 ex 206*

Get Me to the Church on Time *example* 4.4 ex 46, *9.3 ex 46*

Ginger Beer *example* 18.29 ex 365, *9.11 ex 365*

`given`, in calculations with equations 15.18

`giving it to` *action* *7.4*

global constants *see* constant values

global variables *see* variables: global

Glulx 2.14, 23.3

- extensions for Glulx only 27.7
- memory limits 2.14, 20.2, 21.10
- no character set limits 20.4
- no support for Vorple 25.12
- numeric limits 11.10, 15.2, 15.10, 15.14
- real numbers available 15.2
- required for multimedia 23.1, *12.5*

`Glulx resource ID of` *(figure name)* *phrase* 23.10

`Glulx resource ID of` *(sound name)* *phrase* 23.10

Go button *user interface* 1.4

Goat-Cheese and Sage Chicken *example* 16.14 ex 280, *11.4 ex 280*

God's own teacup 16.18

`going` *action* 7.13, *6.9*

- default failure response 7.13
- `going by` *action* 7.14
- `going from` *action* 7.13
- `going nowhere` *action* 7.13
- `going through` *action* 7.14

`going to` *action* 7.13

`going with` *action* 7.14

other characters moving around *7.13*

rules for going by vehicles 7.14

rules for going through doors 7.14

rules for going to and from places 7.13

rules for going with pushed things 7.14

`going on` *(activity)* *adjective* 18.2

GONEAR *testing command* 24.4

Gopher-wood *example* 3.18 ex 32, *7.1 ex 32*

The Gorge at George *example* 17.21 ex 330, *7.6 ex 330*

grammar of player's command *see* understanding

grammar tokens 17.1, 17.4, *6.2*

defining 17.13

defining in Inform 6 27.23

for kinds of value 17.9, 17.11

for numbers 17.6, 17.9

for rooms 17.7, *5.5*

for text 17.5

for things not in scope 17.7, *5.5*

producing values 17.14

Graph-theory view of relations *example* 13.16 ex 244, *1.4 ex 244*

graphics *see* figures

`greater than` *relation verb* 15.5

Grilling *example* 7.2 ex 84, *10.9 ex 84*

`group` *(description of objects)* `together` *phrase* 18.13

`group` *(description of objects)* `together as` *(text)* *phrase* 18.13

`group` *(description of objects)* `together giving articles` *phrase* 18.13

group relations 13.7

`grouping together something` *activity* 18.14

Guided Tour *example* 18.23 ex 351, *12.2 ex 351*

H

`had been`, for past conditions 9.13

`handled` *(thing)* *property* 3.11

The Hang of Thursdays *example* 11.18 ex 183, *4.1 ex 183*

`happened -- if` *(scene)* `has happened` *phrase* 10.4

`happening` *(scene)* *adjective* 10.4

Happy Hour *example* 18.25 ex 356, *7.16 ex 356*

Hardy, Thomas 4.5

`has` *relation verb* *see* possession

`has been`, for past conditions 9.13

`has`, for defining properties 4.8

Hatless *example* 8.11 ex 126, *11.1 ex 126*

A Haughty Spirit *example* 11.17 ex 181, *3.6 ex 181*

"[have]" *say phrase* 14.2

"[haven't]" *say phrase* 14.8

Hays Code *example* 18.9 ex 338, *9.3 ex 338*

Hayseed *example* 7.4 ex 89, *3.5 ex 89*

Hazel, Philip 20.6

headings 2.5, 2.6, 2.9, 25.6

in extension documentation 27.12

in extensions 27.9, 27.10

headline *bibliographic data* 18.37, 25.2

heat

on fire *10.8*

too hot to touch *10.9*

`held by -- first thing held by` *(object) phrase* 8.17

`held by -- next thing held after` *(object) phrase* 8.17

`held in` *relation verb* *see* containment

`held` *(thing) adjective* 3.20, 8.13

help systems *11.3*

HER, in player commands 17.18, *2.2*

hiding things carried by other characters 3.24, 18.9

hiding things from room descriptions by making them scenery 3.8

hiding things under other things *6.6*

Higher Calling *example* 8.15 ex 128, *3.5 ex 128*

HIM, in player commands 17.18, *2.2*

hint systems *11.3*

historical conditions 9.13

times 9.14

turns 9.15

warning concerning variables and other varying qualities 9.13, 9.15

History Lab *example* 14.9 ex 250, *2.1 ex 250*

Hohmann Transfer *example* 18.22 ex 348, *3.7 ex 348*

hold-all 3.21

`holder of` *(object)* *phrase* 8.17

Hot Glass Looks Like Cold Glass *example* 17.21 ex 331, *10.9 ex 331*

Hotel Stechelberg *example* 6.14 ex 79, *3.4 ex 79*

Hour of the Wren *example* 9.11 ex 146, *4.3 ex 146*

(number) `hours` *phrase* 9.8

`hours part of` *(time)* *phrase* 9.10

Hover *example* 3.16 ex 29, *8.1 ex 29*

Hudsucker Industries *example* 18.11 ex 342, *9.2 ex 342*

A Humble Wayside Flower *example* 13.12 ex 237, *7.1 ex 237*

Hunter, Andrew 1.2

Hymenaeus *example* 17.15 ex 312, *3.7 ex 312*

`hyperbolic arccosine of` *(real number)* *phrase* 15.7

`hyperbolic arcsine of` *(real number)* *phrase* 15.7

`hyperbolic arctangent of` *(real number)* *phrase* 15.7

`hyperbolic cosine of` *(real number)* *phrase* 15.7

`hyperbolic sine of` *(real number)* *phrase* 15.7

`hyperbolic tangent of` *(real number)* *phrase* 15.7

The Hypnotist of Blois *example* 12.4 ex 186, *7.14 ex 186*

I

I Didn't Come All The Way From Great Portland Street *example* 21.11 ex 435, *6.14 ex 435*

Ibid. *example* 17.6 ex 300, *12.3 ex 300*

IDE interface *see* user interface

Identity Theft *example* 20.7 ex 416, *5.2 ex 416*

`if` *(a condition)* *phrase* 2.2, 8.11, 8.13, 11.6

with generic conditions 8.14

with `otherwise/else` 11.8

`if` *(a condition)* `,` *(a phrase)* *phrase* 11.6

`if a paragraph break is pending` *phrase* 5.8

`if a random chance of` *(number)* `in` *(number)* `succeeds` *phrase* 8.18

`if` *(action)* `involves` *(object)* *phrase* 12.20

`if action requires a carried noun` *phrase* 12.7

`if action requires a carried second noun` *phrase* 12.7

`if action requires a touchable noun` *phrase* 12.7

`if action requires a touchable second noun` *phrase* 12.7

if action requires light phrase 12.7

IF Archive web site 26.3, 26.4, 26.6

if (external file) exists phrase 23.13

if handling (activity) activity phrase 18.7

if handling (activity on values) activity with (value) phrase 18.7

if in darkness phrase 11.5

If It Hadn't Been For... example 16.9 ex 273, *9.12 ex 273*

if (name of kind) relates to (value) by (relation of values to values) phrase 13.13

if player consents phrase 11.5

if ready to read (external file) phrase 23.15

if rule failed phrase 19.11

if rule succeeded phrase 19.11

if (scene) has ended phrase 10.4

if (scene) has happened phrase 10.4

if (scene) has not ended phrase 10.4

if (scene) has not happened phrase 10.4

if (snippet) does not include (topic) phrase 18.33

if (snippet) does not match (topic) phrase 18.33

if (snippet) includes (topic) phrase 18.33

if (snippet) matches (topic) phrase 18.33

if story has ended phrase 9.4

if story has ended finally *phrase* 9.4

if story has not ended *phrase* 9.4

if story has not ended finally *phrase* 9.4

if *(text)* exactly matches the regular expression *(text)* *phrase* 20.6

if *(text)* exactly matches the text *(text)* *phrase* 20.5

if *(text)* is in lower case *phrase* 20.4

if *(text)* is in upper case *phrase* 20.4

if *(text)* matches the regular expression *(text)* *phrase* 20.6

if *(text)* matches the text *(text)* *phrase* 20.5

if there is *(a table entry)* *phrase* 16.7

if there is no *(a table entry)* *phrase* 16.7

if *(time)* is after *(time)* *phrase* 9.9

if *(time)* is before *(time)* *phrase* 9.9

if *(value)* is *phrase* 11.8

if *(value)* is listed in *(list of values)* *phrase* 21.4

if *(value)* is not listed in *(list of values)* *phrase* 21.4

if *(value)* matches *(description of values)* *phrase* 22.2

if *(value)* relates to *(name of kind)* by *(relation of values to values)* *phrase* 13.13

IFDB *web site* 25.8, 26.3, 26.4, 26.6

IFID *bibliographic data* 23.12, 23.15, 25.5, 26.6, *11.5*

ifwiki *web site* 26.5, 26.7, *1.2*

Igpay Atinlay *example* 20.8 ex 421, *2.3 ex 421*

images *see* figures

immobile things

`fixed in place` 3.6, 8.7

moving anyway 8.7

`scenery` 3.8

supporters made automatically immobile 3.7

`implicitly taking something` *activity* 12.17, 18.34, *6.8*

`improper-named / proper-named` *(thing)* *property* 3.18

`in` *relation verb* *see* containment

In Fire or in Flood *example* 19.8 ex 401, *10.8 ex 401*

`in groups`, in defining relations 13.7

`in`, location conditions for rules 7.11, 7.13

(text) `in lower case` *phrase* 20.4

(text) `in sentence case` *phrase* 20.4

`in the presence of` 7.12, 18.29

(rule) `in` *(time)* `from now` *phrase* 9.11

(text) `in title case` *phrase* 20.4

(rule) `in` *(number)* `turn/turns from now` *phrase* 9.11

(text) `in upper case` *phrase* 20.4

`inactive / active` *(use option)* *adjective* 27.18

Inanimate Listeners *extension* *7.7*, *9.6*

`include` *(extension)* `by` *(extension author)* *assertion* 2.11

incorporation *relation* 3.23, 4.15, 8.13, 13.4

`increase` *(a stored value)* `by` *(value)* *phrase* 8.12, 9.2

`increment` *(a stored value)* *phrase* 8.12

`indefinite article of` *(object)* *property* 3.18

indefinite articles

- when creating things 3.18
- when displaying lists 5.5, 21.3
- when displaying things 5.3

indentation *of source text* 11.7, 11.9

index *of source text* *see* headings

index map 3.2, 3.4, 3.26, *3.2*

- customizing for release 25.18, *13.2*

Index panel *user interface* 1.6

- Actions page 12.5, 12.21, 17.1, 17.2, 17.3, 24.3
- checking story correctness 24.1
- Contents page 2.6, 23.5, 27.10
- Kinds page 4.1, 4.6, 4.8, 4.11, 4.12, 27.10
- Map page 2.7, 3.4, 3.26, 25.18
- Phrases page 2.2, 11.2
- Rules page 19.6, 19.16
- Scenes page 10.3, 10.9

indexed text, now the same as text *see* text

indirect containment 3.25, 6.11

Indirection *example* 17.1 ex 286, *3.2 ex 286*

`inedible / edible` *(thing)* *property* 3.22, *9.1*

Infiltration *example* 9.14 ex 152, *3.1 ex 152*

`infinite / finite` *(real number)* *adjective* 15.3

Infocom 2.12, 9.3, 25.7, *12.2*, *12.3*

inform7.com *web site* 2.15, 27.3

Inform 6 1.1, 2.14, 27.14

Inform 6 Designer's Manual 18.29, 27.21, 27.23, 27.27

Inform 6 equivalent

- attributes 4.7
- `BeforeParsing` 18.33
- `box` 5.12
- `break` 11.12
- `child` 8.17
- classes 4.1
- `clothing` 3.20
- code blocks 11.7
- constants 4.13
- `continue` 11.12

daemons 9.5

darkness 12.19

each_turn 9.5

for 11.10

functions 11.2

functions with optional parameters 11.14

functions with parameters 11.3

global variable 4.12

IndirectlyContains 3.25

initial 3.11

local variables 11.15

meta 12.15

objectloop 11.11

parent 8.17

procedures 11.2

properties 4.8

return 11.13

return value 11.17

rtrue/rfalse 11.16

SACK_OBJECT 3.21

scope rules 12.16, 12.18, 18.29

sibling 8.17

`switch/case` 11.8

`switchable` 3.14

synonyms 12.7

timers 9.11

verbs 12.7

`while` 11.9

`with_key` 3.13

Inform 6 inclusions 27.14

actions 27.22

adjectives 27.24

classes 27.21

global variables 27.22

grammar tokens 27.23

Inform 6 template layer 27.26, *6.3*

long sections of code 27.19

objects 27.21

phrase elements 27.28, 27.29, 27.30

phrase options 27.17

phrases 27.15

phrases to decide a value 27.16

phrases to decide whether/if 27.16

properties 27.21, 27.22

rules 27.20

understanding grammar 27.23

use options 27.18

information *see* knowledge (in story world)

Ingold, Jon *1.2*

`initial appearance of` *(thing) property* 3.11, 8.5

initial location of player 3.1

initial state of the world 2.1, 3.1

scenes that begin at the beginning 10.6

setting in `when play begins` 9.1, *11.1*

`initially`, in defining global variables 4.12

`inserting it into` *action* *6.8*

`inside` *relation verb* *see* containment

`inside from` 3.2

`instead of` *(action) rulebook* 7.2, 7.3

as failure of persuasion 12.4

guidelines for use 12.21

in action processing sequence 12.2, *6.3*

not used for `out of world` actions 12.15

`instead of`, replacing rules 19.4

`instead`, to stop the action 7.3, 12.9, 19.11, *6.3*

instructing other characters 12.3, *7.14*

`insufficient / sufficient light, there is insufficient / sufficient light` *(visibility rules)* *outcome phrase* 12.19

integers *see* numbers

interface *see* user interface

Interrogation *example* 13.4 ex 225, *10.10 ex 225*

intfiction forum *web site* 24.8, 26.3

introduction *see* initial state of the world

Introduction to Juggling *example* 16.16 ex 283, *9.4 ex 283*

introductory booklet and postcard 25.9

`inventory details -- printing inventory details of something` *activity* 18.17

`inventory -- taking inventory` *action* *6.7*

`invisible / visible` *(thing)* *adjective* 6.13

IPA *example* 9.6 ex 141, *4.3 ex 141*

IQ Test *example* 12.6 ex 189, *7.15 ex 189*

`is` 2.1, 13.1, 13.2, 13.8

equality relation *see* equality

Ish. *example* 17.5 ex 296, *11.3 ex 296*

`issuing the response text of something` *activity* 18.15

`"[it]"` *say phrase* 14.5

`it does / it does not` *(deciding whether all includes)* *outcome phrase* 18.36

IT, in player commands 17.18, *2.2*

it, in source text 11.16

it is likely / possible / unlikely *(does the player mean) outcome phrase* 17.19

item described — *object global variable* 17.17, *2.2*

J

Jamaica 1688 *example* 18.38 ex 385, *11.6 ex 385*

Jerome, Jerome K. 2.5, 25.2

Jerz, Dennis 26.9

Jokers Wild *example* 16.11 ex 275, *9.5 ex 275*

K

Kesserich, Sonja 1.2

keyword-style parsing *6.18*

Kinder, David 1.2

kinds 3.5, 4.1, 22.1

catalogue

animal 3.17

backdrop 3.9; *see also* backdrops

container 3.5; *see also* containers

device 3.14, *9.8*, *10.7*

direction 3.5, 3.26

door 3.12; *see also* doors

man 3.17

person 3.17; *see also* person

player's holdall 3.21

region 3.4; *see also* regions

room 3.5; *see also* rooms

supporter 3.5; *see also* supporters

thing 3.5; *see also* things

vehicle 3.16, *8.1*

woman 3.17

default properties 4.3

default values of kinds 4.8, 4.11, 4.12, 22.4

defining 4.1

in Inform 6 27.21

kinds of thing 4.1, 4.2, 4.4

kinds of value 4.5, 4.9, 4.10

defining with a table 16.17

multiple kinds with a table 16.16

documenting in extensions 27.10

enumerated values 4.5, 4.9, 4.10

hierarchy of kinds 4.1

in phrase parameters 11.3, 22.8

kind variables 22.1, 22.7

must define before using 4.2

of action 7.15

of description 22.2

of kinds 22.1, 22.6

of list 21.1, 21.2

of phrase 22.3

default values for 22.4

of relation 13.14

of thing 4.1

of value 4.1, 4.5

of values with units of measure 15.8

uses of new kinds 4.2

Kinds page of Index panel *user interface* 4.1, 4.6, 4.8, 4.11, 4.12, 27.10

Kiwi *example* 18.28 ex 361, *8.4 ex 361*

Kleene, Stephen 20.6

Knight, Cedric *6.17*

knowledge (in story world)

other characters' knowledge *7.11*

player's knowledge *5.4*, *5.5*

Knuth, Donald 1.1

Kyoto *example* 19.11 ex 404, *10.4 ex 404*

L

Labyrinth of Ghosts *example* 23.13 ex 441, *11.6 ex 441*

Lakeside Living *example* 17.22 ex 333, *10.2 ex 333*

Lanista 1 *example* 8.18 ex 130, *7.5 ex 130*

Lanista 2 *example* 17.3 ex 293, *7.5 ex 293*

`last`, listed last in rulebook 19.4, 19.7

`last value of` *(name of kind)* *phrase* 11.18

`latest parser error` — *command parser error* *global variable* 18.35

Latin Lessons *example* 18.32 ex 371, *7.14 ex 371*

Latris Theon *example* 12.3 ex 185, *7.13 ex 185*

Laura *example* 3.11 ex 19, *2.2 ex 19*

laws for sorting rulebooks 19.6, 19.16

Le Guin, Ursula K. 3.12

Lead Cuts Paper *example* 15.12 ex 259, *10.5 ex 259*

`leading zeroes`, when defining units 15.15

Lean and Hungry *example* 6.13 ex 76, *7.2 ex 76*

Lebling, Dave 2.5, 2.13

The Left Hand of Autumn *example* 17.4 ex 295, *6.5 ex 295*

`left hand status line` — *text* *global variable* 8.3, 18.23, *12.2*

Leinonen, Juhana 1.2, 24.6, 25.12

Lemonade *example* 15.19 ex 266, *10.2 ex 266*

Leopard-skin *example* 21.10 ex 430, *6.14 ex 430*

`less than` *relation verb* 15.5

`less than`, in descriptions 6.16

`let` *(a name not so far used)* `be` *(description of relations of values to values)* *phrase* 13.15

let *(a name not so far used)* be given by *(equation name)* *phrase* 15.18

let *(a name not so far used)* be *(name of kind)* *phrase* 11.15

let *(a name not so far used)* be *(value)* *phrase* 11.15

let *(a temporary named value)* be given by *(equation name)* *phrase* 15.18

let *(a temporary named value)* be *(value)* *phrase* 11.15

Lethal Concentration 1 *example* 15.8 ex 255, *10.1 ex 255*

Lethal Concentration 2 *example* 15.8 ex 257, *10.1 ex 257*

letters *see* characters (letters)

Lies *example* 17.13 ex 310, *6.10 ex 310*

light 3.15, 6.13, *3.7*

- lack of *see* darkness
- printing the announcement of light *activity* 18.20
- requirement in defining actions 12.7, 12.17
- rules to override sufficiency of 12.19

lighted / dark *(room)* *property* 3.15, *3.7*

likely / possible / unlikely, it is likely / possible / unlikely *(does the player mean)* *outcome phrase* 17.19

limits

- for extending lists 21.10
- for manipulating text 20.2
- of Inform 6 compiler 2.14
- of numeric values 11.10, 15.2

of numeric values of units 15.10, 15.14

of scene changing mechanism 10.9

of story file 2.14

on changing lists being repeated through 21.4

line breaks 5.6, 5.8

counting lines in text 20.3

in formatting tables 16.1

produced by sentence-ending punctuation 2.3, 5.2, 5.8

`line number` *(number)* `in` *(text)* *phrase* 20.3

liquids (implementing) *10.2*

`"[list of` *(description)*`]"` *say phrase* 6.1

`list of` *(description of values)* *phrase* 21.6

`list of` *(name of kind)* `that/which/who relate to` *(value)* `by` *(relation of values to values)* *phrase* 13.13

`list of` *(name of kind)* `that/which/whom` *(relation of values to values)* `relates` *phrase* 13.13

`list of` *(name of kind)* `that/which/whom` *(relation of values to values)* `relates to` *phrase* 13.13

`list of` *(name of kind)* `that/which/whom` *(value)* `relates to by` *(relation of values to values)* *phrase* 13.13

`list of` *(name of kind)* `to which/whom` *(relation of values to values)* `relates` *phrase* 13.13

`list of` *(name of kind)* `to which/whom` *(value)* `relates by` *(relation of values to values)* *phrase* 13.13

`list the contents of` *(object)* *phrase* 11.14

`listed in / before / after / instead of`, for placing rules in rulebooks 19.4

`listening to` *action* 7.7, *3.8*

`listing contents of something` *activity* 18.13

`listing group size` — *number* *global variable* 18.12, 18.14

`listing nondescript items of something` *activity* 18.25, *3.1*

listing rules

- for grouping together 18.14
- for listing contents 18.13

lists 13.13, 21.1

- adding entries 21.1, 21.5
- changing entries 21.9
- changing length 21.10
- constant lists 21.2
- creating 21.2
  - by description 21.6, 21.7
- displaying 5.5, 21.3
- `empty / non-empty` *adjective* 21.1, 21.2
- extending 21.10
- filtering (with a phrase or description) 22.5
- finding entries 21.4, 21.9
- finding length 21.1, 21.9
- mapping (with a phrase) 22.5

memory limits 21.10

reducing (with a phrase, to a value) 22.5

removing entries 21.5

repeating through 21.4

reversing 21.8

rotating 21.8

sorting 21.8

the multiple object list 17.20, *6.15*

truncating 21.10

variations on use of lists 21.11

Z-machine limitations 21.10

`lit / unlit` *(thing) property* 3.15, *3.7*

local variables *see* temporary variables

`locale description -- printing the locale description of something` *activity* 18.26

`locale paragraph -- printing a locale paragraph about something` *activity* 18.28, *3.1*

`location of` *(object) phrase* 3.25

`location` — *object global variable* 3.25, 4.12

locations *see* rooms

`lockable` *(container/door) property* 3.13, *9.2*

`locked / unlocked` *(container/door) property* 3.13, *9.2*

Locksmith *extension* 2.10, 2.11, *3.5*, *9.2*

`logarithm to base` *(number)* `of` *(real number)* *phrase* 15.6

Lollipop Guild *example* 18.34 ex 379, *6.8 ex 379*

`looking` *action* 12.19, *3.1*, *6.4*

- as part of going *6.9*

loops

- counted 11.10
- in phrases 11.9
- interrupting 11.12
- through objects 11.11
- through values 11.10

losing the game 9.4

losing the story 18.38, 18.39, *11.6*

Low Light *example* 18.27 ex 359, *3.1 ex 359*

`lower case --` *(text)* `in lower case` *phrase* 20.4

Lucy *example* 7.6 ex 92, *6.14 ex 92*

Lugubrious Pete's Delicatessen *example* 21.11 ex 436, *7.16 ex 436*

M

M. Melmoth's Duel *example* 11.15 ex 178, *2.1 ex 178*

machines (implementing) *9.8*, *10.7*

magic words 24.4, *6.13*

Magneto's Revenge *example* 12.18 ex 215, *10.10 ex 215*

magnets (implementing) *10.7*

`make no decision` *phrase* 19.11

`male / female / neuter` *(person)* *property* 3.17

man *kind* 3.17, 4.1

The Man of Steel *example* 12.11 ex 203, *7.15 ex 203*

The Man of Steel Excuses Himself *example* 12.13 ex 208, *7.15 ex 208*

map *see* index map

Map page of Index panel *user interface* 2.7, 3.4, 3.26, 25.18

mapping *(direction)* *relation* 6.14

`mark` *(external file)* `as not ready to read` *phrase* 23.15

`mark` *(external file)* `as ready to read` *phrase* 23.15

`marked for listing / unmarked for listing` *(thing)* *property* 18.25, *3.1*

Masochism Deli *example* 17.19 ex 326, *10.9 ex 326*

mass nouns 3.18

`matched as`, in describing action variables 12.10

`matched text` — *snippet* *global variable* 18.33

`matching key of` *(container/door)* *property* 3.13

materials folder

- cover art 25.8
- feelies 25.7
- figures 23.4
- introductory booklet / postcard 25.9

map 25.18, *13.2*

pre-existing story file 25.15

project-specific extensions 27.3

released story file 25.6

sounds 23.8

source text 25.17

walkthrough 25.16

web-playable story 25.11, 26.5

web pages for the story 25.10, 26.5

templates 25.13, 25.14

Mathematical view of relations *example* 13.16 ex 243, *1.4 ex 243*

maths *see* calculation

Matreshka *example* 11.7 ex 172, *6.6 ex 172*

Mattress King *example* 7.14 ex 105, *6.9 ex 105*

`maximum score` — *number* *global variable* 9.2, 9.3, *11.4*

`maximum text length` *use option* 20.2

`maximum things understood at once` *use option* 4.14

`"[may]"` *say phrase* 14.7

`"[mayn't]"` *say phrase* 14.7

Mazaitis, Paul 1.2

McGrew, Jesse 27.4, *1.2*

McKee, Kate *1.2*

meaning *relation* 14.9

`meaningful / meaningless` *(verb)* *adjective* 14.9

`means`, defining sentence verbs

`the verb to be` *(P)* `means` *(relation)* *assertion* 13.10

`the verb` *(V)* `means` *(property)* *assertion* 15.12

`the verb` *(V)* `means` *(relation)* *assertion* 13.9

Meet Market *example* 13.13 ex 238, *7.1 ex 238*

`member of` *(description)* 22.2

`memory economy` *use option* 2.14

memory limits 2.14

for extending lists 21.10

for manipulating text 20.2

memory of player *see* knowledge (in story world)

men *kind* *see* man

`mentioned / unmentioned` *(thing)* *property* 18.24, 18.26, 18.28, *3.1*

Merlin *example* 16.13 ex 277, *5.4 ex 277*

metadata *see* bibliographic data

Meteoric I and II *example* 8.7 ex 119, *4.4 ex 119*

Metric Units *extension* 15.13, 15.18, 15.20

Midsummer Day *example* 3.5 ex 12, *1.1 ex 12*

`"[might]"` *say phrase* 14.7

"[mightn't]" *say phrase* 14.7

Mimicry *example* 7.15 ex 110, *7.6 ex 110*

Ming Vase *example* 7.10 ex 97, *10.4 ex 97*

Minimal Movement *example* 18.32 ex 372, *6.9 ex 372*

(arithmetic value) `minus` *(arithmetic value)* *phrase* 15.5

`minus infinity` — *real number* *constant* 15.3

(number) `minutes` *phrase* 9.8

`minutes part of` *(time)* *phrase* 9.10

Mirror, Mirror *example* 20.7 ex 417, *9.12 ex 417*

Misadventure *example* 17.10 ex 306, *6.9 ex 306*

`missing noun -- supplying a missing noun/second noun` *activity* 17.1, 18.32

mistakes, in the player's command 17.21, 17.22, *5.3*

Mistress of Animals *example* 6.14 ex 77, *7.13 ex 77*

Mistype *extension* *6.17*

`modal / non-modal` *(verb)* *adjective* 14.7, 14.9

Model Shop *example* 4.15 ex 57, *9.8 ex 57*

Modern Conveniences *example* 27.5 ex 449, *8.5 ex 449*

Money for Nothing *example* 15.19 ex 265, *9.4 ex 265*

money (implementing) *9.4*

monospaced text 5.9, *12.1*

Montfort, Nick *1.2*

Monty Python 11.7

`more than`, in descriptions 6.16

Moriarty, Brian 5.12

Morning After *example* 7.5 ex 90, *6.8 ex 90*

`most`, in descriptions 6.15

`move` *(backdrop)* `to` *(region)* *phrase* 8.8

`move` *(object)* `backdrop to all` *(description of objects)* *phrase* 8.8

`move` *(object)* `to` *(object)* *phrase* 8.7

Mr. Burns' Repast *example* 20.8 ex 422, *2.3 ex 422*

MRE *example* 9.11 ex 143, *9.1 ex 143*

multiple copies of things *see* duplicates

`multiple object list` *phrase* 17.20, *6.15*

(arithmetic value) `multiplied by` *(arithmetic value)* *phrase* 15.5

Murder on the Orient Express *example* 13.16 ex 241, *7.11 ex 241*

Musante, Mark 1.2

music *see* sounds

`"[must]"` *say phrase* 14.7

`"[mustn't]"` *say phrase* 14.7

Mutt's Adventure *example* 9.2 ex 136, *11.4 ex 136*

N

Nameless *example* 17.5 ex 297, *7.7 ex 297*

names

for things named generally 8.15

of characters *7.1*

of phrases 22.3

of rules 19.2, 19.3, 19.7

`printed name of` *(object)* *property* 8.5

`printing the name of a dark room` *activity* 18.21

`printing the name of something` *activity* 18.10

`printing the plural name of something` *activity* 18.11

NaN (not a number) *see* nonexistent / existent (real number)

Narrative Register *example* 14.3 ex 248, *2.1 ex 248*

narrative viewpoint, of standard responses 14.1, *5.6*

`natural/-- logarithm of` *(real number)* *phrase* 15.6

natural language, really understanding 2.16

`nearest whole number --` *(real number)* `to the nearest whole number` *phrase* 15.3

`nearest --` *(arithmetic value)* `to the nearest` *(arithmetic value)* *phrase* 9.8, 15.5

`negative / positive` *(number)* *adjective* 6.5

Neighborhood Watch *example* 3.13 ex 24, *3.5 ex 24*

Nelson, Toby 1.2

`neuter / male / female` *(person)* *property* 3.17

`never / always`, in assertions 4.3

Newcomb, Ron 1.2

`next` *phrase* 11.12

`next step via` *(relation of values to values)* `from` *(object)* `to` *(object)* *phrase* 13.11

`next thing held after` *(object)* *phrase* 8.17

Nickel and Dimed *example* 15.17 ex 262, *9.4 ex 262*

Night and Day *example* 10.7 ex 164, *3.9 ex 164*

The Night Before *example* 4.15 ex 59, *5.1 ex 59*

Night Sky *example* 9.12 ex 147, *3.1 ex 147*

Nine AM Appointment *example* 19.2 ex 388, *6.11 ex 388*

`no` *phrase* 11.16

`no`, in descriptions 6.15

No Place Like Home *example* 9.2 ex 137, *11.4 ex 137*

NO, querying yes/no 11.5

No Relation *example* 7.14 ex 104, *8.1 ex 104*

NO, responding to a character *7.7*

`nobody`, testing that no such person exists 6.10

Noisemaking *example* 12.9 ex 197, *7.3 ex 197*

Noisy Cricket *example* 16.12 ex 276, *10.2 ex 276*

`non-empty / empty` *adjective*

- for activities 18.5
- for lists 21.1, 21.2
- for relations 13.14
- for rulebooks 19.8
- for tables 16.9

for texts 6.5, 20.3

non-existence

removing things from play 8.10

starting things out of play 3.25, 6.10, 8.10

testing that no such thing exists 6.10, 6.12

`non-modal / modal` *(verb)* *adjective* 14.9

`non-recurring / recurring` *(scene)* *property* 10.4, 10.7

`nondescript items -- listing nondescript items of something` *activity* 18.25, *3.1*

`none`, in descriptions 6.15

`nonexistent / existent` *(real number)* *adjective* 15.3

North by Northwest *example* 18.33 ex 375, *2.2 ex 375*

Northstar *example* 20.8 ex 423, *7.14 ex 423*

`not for release` 2.9, 24.5, 25.6, *13.1*

`not listed in`, for removing rules from rulebooks 19.4

`notable locale objects -- choosing notable locale objects for something` *activity* 18.27

`nothing`

as a value 6.12

`do nothing` *phrase* 11.1

in removing room connections 8.5

in understanding actions 17.3

testing that no such thing exists 6.10, 6.12

NOTIFY ON/OFF 9.2, *11.4*, *11.5*

`noun part of` *(action) phrase* 12.20

`noun` — *object global variable* 7.10

nouns, in descriptions 6.2

nouns, reversed, in understanding actions 17.2

`now` *(a condition) phrase* 2.2, 8.1, 8.11

changing either/or properties 8.4

changing global variables 8.1

changing list entries 21.9

changing table entries 16.4

changing temporary variables 11.15

changing the locations of backdrops 8.8

changing the locations of things 8.7, 8.10

changing the player's identity 8.9, *5.6*

changing the player's location 8.9

changing value properties 8.5

for arbitrary conditions 8.11

`now --` *(rule)* `in` *(number)* `turn/turns from now` *phrase* 9.11

`now --` *(rule)* `in` *(time)* `from now` *phrase* 9.11

`nowhere`

in removing room connections 3.3, 8.5

placing backdrops 3.9, 8.8

placing things 8.10

rules for going nowhere 7.13

NPC *see* characters (people)

`[number]` *grammar token* 17.6, 17.9

`number of blank rows in/from` *(table name)* *phrase* 16.10

`number of characters in` *(text)* *phrase* 20.3

`number of` *(description of values)* *phrase* 6.1, 8.16

`number of entries in/of` *(list of values)* *phrase* 21.1, 21.9

`number of filled rows in/from` *(table name)* *phrase* 16.10

`number of lines in` *(text)* *phrase* 20.3

`number of moves from` *(object)* `to` *(object)* *phrase* 6.14

`number of moves from` *(object)* `to` *(object)* `through` *(description of objects)* *phrase* 6.14

`number of paragraphs in` *(text)* *phrase* 20.3

`number of punctuated words in` *(text)* *phrase* 20.3

`number of rows in/from` *(table name)* *phrase* 16.2

`number of steps via` *(relation of values to values)* `from` *(object)* `to` *(object)* *phrase* 13.11

`number of times` *(text)* `matches the regular expression` *(text)* *phrase* 20.6

`number of times` *(text)* `matches the text` *(text)* *phrase* 20.5

`number of unpunctuated words in` *(text)* *phrase* 20.3

`number of words in` *(text)* *phrase* 20.3

Number Study *example* 13.14 ex 240, *10.11 ex 240*

`number understood` — *number global variable* 17.9

Numberless *example* 11.8 ex 174, *2.1 ex 174*

numbers 15.1, 15.2

- adjectives comparing magnitude 6.7, 6.8
- adjectives describing magnitude 6.6
- arithmetic 15.5
- calculation with *see* calculation
- conversion to/from real numbers 15.3
- converting to times 9.10
- displaying 5.3, 5.4
- given by equations 15.18
- limits of numeric values 11.10, 15.2
- limits of numeric values of units 15.10, 15.14
- of things in descriptions 6.16
- `printing a number of something` *activity* 18.12
- random numbers *see* randomness
- real numbers 15.2
  - conversion to/from numbers 15.3
  - displaying 15.4
  - exponential functions 15.6

infinite and nonexistent 15.3

trigonometric functions 15.7

understanding 17.6, 17.9

units of measure *see* units of measure

Nyman, Jeff *1.2*

O

object loops 11.11

Object Response Tests *extension* 24.6

objects *see* things

`odd / even` *(number)* *adjective* 4.6, 6.5

Odin *example* 4.3 ex 44, *6.5 ex 44*

Odyssey *example* 16.10 ex 274, *7.13 ex 274*

`of`, for properties 4.9

`off-stage / on-stage` *(thing)* *adjective* 8.10

Olfactory Settings *example* 14.6 ex 249, *2.1 ex 249*

`omit contents in listing` *phrase* 18.10

omitting code, for release version 2.9, 24.5, 25.6, *13.1*

`on-stage / off-stage` *(thing)* *adjective* 8.10

`on` *relation verb* *see* support

`on top of` *relation verb* *see* support

`once`, counting conditions 9.14

`one-to-one` *(relation)* *adjective* 13.14

`one-to-various` *(relation)* *adjective* 13.14

one-way connections 3.3

One of Those Mornings *example* 17.7 ex 301, *5.4 ex 301*

One Short Plank *example* 7.14 ex 106, *3.5 ex 106*

`only`, counting conditions 9.14, 9.15

Only You... *example* 6.5 ex 74, *10.1 ex 74*

`opaque / transparent` *(container)* *property* 3.6, 3.15, *9.2*

`open / closed` *(container/door)* *property* 3.6, 3.13, 3.15, *9.2*

`openable / unopenable` *(container)* *property* 3.6, *9.2*

`opposite of` *(direction)* *property* 3.26

Options.txt 2.13

options, for phrases 11.14

`or more / less` 6.6, 6.7, 6.8

Orange Cones *example* 8.8 ex 120, *3.9 ex 120*

ordering *see* precedence

ordering other characters 12.3, *7.14*

Originals *example* 18.31 ex 369, *9.12 ex 369*

other people *see* characters (people)

`other side of` *(door)* `from` *(room)* *phrase* 3.12

`[other things]` *grammar token* 17.4

`otherwise` *(a phrase)* *phrase* 11.8

`otherwise if` *(a condition)* *phrase* 11.8

`otherwise unless` *(a condition)* *phrase* 11.8

Otranto *example* 13.7 ex 228, *10.6 ex 228*

`"[our]"` *say phrase* 14.4

out of play 3.25, 6.10, 8.10

`out of world` 12.15, 12.21

`outcome of the rulebook` *phrase* 19.12

outline *of source text* 2.6

`outside from` 3.2

Owen's Law *example* 11.16 ex 179, *6.9 ex 179*

Oyster Wide Shut *example* 21.3 ex 425, *6.7 ex 425*

P

Paddington *example* 12.9 ex 195, *10.4 ex 195*

Pages *example* 17.9 ex 303, *9.6 ex 303*

paint (implementing) *9.7*

Palette *example* 17.11 ex 308, *9.7 ex 308*

Panache *example* 10.8 ex 166, *11.4 ex 166*

paragraph breaks 5.6, 5.8

- counting paragraphs in text 20.3
- for confusing the normal display of objects 18.24
- for fooling persuasion refusal 12.4
- with vertical stroke 2.3

`paragraph number` *(number)* `in` *(text)* *phrase* 20.3

`paragraph -- writing a paragraph about` *activity* 18.24, *3.1*

parameters

of phrases 11.3

kind values 22.8

of rulebooks 19.9

parentheses *see* punctuation: brackets

`parser's choice -- clarifying the parser's choice of something` *activity* 18.30

parser error messages 18.35, *6.17*

parsing *see* understanding

`part` *of source text* 2.5

`part of` *relation verb* *see* incorporation

`part of`, for units with multiple components 15.15

`particular possession` — *thing* *global variable* 3.24, 18.9

`past / present / future tense` 14.1

past tense, of conditions in rules 9.12, 9.13

pathfinding 6.14

arbitrary relations 13.11

Patient Zero *example* 19.8 ex 402, *7.13 ex 402*

pattern matching

phrase definitions 11.3

Peeled *example* 18.29 ex 363, *3.7 ex 363*

Peers *example* 17.15 ex 315, *7.1 ex 315*

people *see* person; characters (people)

perfect tense, of conditions in rules 9.13

period *see* punctuation: full stop

Persephone *example* 5.6 ex 65, *6.7 ex 65*

person *kind* 3.17

carrying capacity 3.19

characters *see* characters (people)

`person asked` — *object global variable* 12.3

`person reaching` — *object global variable* 12.18

`persuasion` *rulebook* 12.4

`persuasion succeeds / fails` *(persuasion rules)* *outcome phrase* 12.4

Peugeot *example* 3.16 ex 27, *8.1 ex 27*

phrases 11.1

as values 22.3

default values 22.4

deciding a condition 11.5, 11.16

deciding a value 11.17

defining 11.2

defining in Inform 6 27.15, 27.16, 27.17

interrupting 11.13

named phrases 22.3

options for small variations 11.14

phrase parameters 11.3

kind values 22.8

precedence of definitions 22.9, 22.10

temporary variables in phrases 11.15

Phrases page of Index panel *user interface* 2.2, 11.2

`pi` — *real number constant* 15.2

pictures *see* figures

Pine 1 *example* 10.2 ex 154, *7.3 ex 154*

Pine 2 *example* 10.5 ex 161, *7.12 ex 161*

Pine 3 *example* 10.7 ex 165, *4.5 ex 165*

Pine 4 *example* 10.8 ex 167, *4.5 ex 167*

Pink or Blue *example* 27.15 ex 451, *5.2 ex 451*

Pizza Prince *example* 18.34 ex 378, *10.3 ex 378*

`place` *(object)* `in scope` *phrase* 18.29

`place the/-- contents of` *(object)* `in scope` *phrase* 18.29

places *see* rooms

`play` *(sound name)* *phrase* 23.8

`player's command` — *snippet global variable* 18.33

player's holdall *kind* 3.21

player

as opposed to the actor 12.3, 12.11

carrying capacity 3.19

changing the identity of the player 8.9, *5.6*

customizing name or traits *5.2*

initial location 3.1

memory *see* knowledge (in story world)

moving the player 8.7, 8.9

plot structure *see* story structure

Plotkin, Andrew 1.2, 25.9

`plural-named / singular-named` *(thing) property* 3.18

`plural of`

in player command 17.8

in source text 4.4

plurals

defining 4.4, *2.2*

displaying 5.4

of units of measure 15.9

plural convention for adaptive pronouns 14.4

`printing the plural name of something` *activity* 18.11

understanding (in commands) 17.8

(arithmetic value) `plus` *(arithmetic value) phrase* 15.5

`plus infinity` — *real number constant* 15.3

A point for never saving the game *example* 12.15 ex 211, *11.2 ex 211*

points *see* scoring

Polarity *example* 7.13 ex 103, *6.9 ex 103*

Police State *example* 12.4 ex 187, *7.3 ex 187*

Politics as Usual *example* 8.3 ex 114, *12.2 ex 114*

Port Royal 1 *example* 3.2 ex 5, *3.2 ex 5*

Port Royal 2 *example* 3.3 ex 8, *3.2 ex 8*

Port Royal 3 *example* 3.4 ex 10, *3.2 ex 10*

Port Royal 4 *example* 16.6 ex 272, *3.6 ex 272*

Port Royal 5 *example* 25.23 ex 447, *13.2 ex 447*

`portable / fixed in place` *(thing)* *property* 3.6, 3.7, 8.7

`positive / negative` *(number)* *adjective* 6.5

possession *relation* 8.13, 13.4

possessions 3.20

- a hold-all for the player's posessions 3.21
- `deciding the concealed possessions of something` *activity* 3.24, 18.9

`"[possessive]"` *say phrase* 14.6

`possible / likely / unlikely, it is possible / likely / unlikely` *(does the player mean)* *outcome phrase* 17.19

Pot of Petunias *example* 17.18 ex 325, *2.2 ex 325*

powers, of numbers 15.6

The Prague Job *example* 10.6 ex 162, *4.2 ex 162*

`preamble optional`, when defining units 15.15

precedence

of commands understood 17.22

of displayed items 18.26, 18.27, *3.1*

of phrase definitions 22.9, 22.10

of rules 19.2, 19.4, 19.6, 19.16

prepositions, for sentence verbs 13.10

`presence of` 7.12, 18.29

`present / past / future tense` 14.1

Princess and the Pea *example* 11.7 ex 173, *8.4 ex 173*

`printed name of` *(object)* *property* 8.5, 8.6, 18.10

`printed plural name of` *(object)* *property* 18.11

`printing a locale paragraph about` *activity* 18.28, *3.1*

`printing a number of something` *activity* 18.12

`printing a parser error` *activity* 18.35

`printing a refusal to act in the dark` *activity* 18.18

`printing an abbreviated room description` 8.7, 8.9

`printing inventory details of something` *activity* 18.17

`printing room description details of something` *activity* 18.16, *3.1*

`printing the announcement of darkness` *activity* 18.19

`printing the announcement of light` *activity* 18.20

`printing the banner text` *activity* 18.37

`printing the description of a dark room` *activity* 18.22

`printing the locale description of something` *activity* 18.26

`printing the name of a dark room` *activity* 18.21

`printing the name of something` *activity* 18.10

`printing the player's obituary` *activity* 18.38

`printing the plural name of something` *activity* 18.11

priority *see* precedence

Priority Lab *example* 18.26 ex 358, *3.1 ex 358*

Prisoner's Dilemma *example* 3.26 ex 40, *3.2 ex 40*

`private / public`, in release elements 25.16, 25.17

`privately-named / publicly-named` *(object)* *adjective* 17.17, *2.2*

problem messages

abject failures of Inform 2.15

for source text 2.4

The Problem of Edith *example* 13.11 ex 235, *7.11 ex 235*

(name of kind) `produced by` *(rule producing values)* *phrase* 19.13

(name of kind) `produced by` *(values based rule producing values)* `for` *(value)* *phrase* 19.13

Prolegomena *example* 18.12 ex 343, *2.1 ex 343*

prompt, changing the prompt 8.2, 9.7

PRONOUNS *11.5*

pronouns 3.17

adaptive contractions in displayed text 14.8

adaptive pronouns in displayed text 14.2, 14.4, 14.5, 14.6

displaying with PRONOUNS *11.5*

HIM/HER, in player commands 17.18, *2.2*

IT, in player commands 17.18, *2.2*

`it`, in source text 11.16

setting pronouns 17.18, *2.2*

`something, anything...` 6.2

THEM, in player commands 17.18, *2.2*

`proper-named / improper-named` *(thing)* *property* 3.18

properties 3.6, 3.7, 4.6

catalogue

`carrying capacity of` *(container/supporter/person)* 3.19, *10.5*

`closed` *(container/door)* 3.6, 3.13, 3.15, *9.2*

`dark` *(room)* 3.15, *3.7*

`described` *(thing)* 3.24

`description of` *(room)* 3.10, 3.11

`description of` *(scene)* 10.2

`description of` *(thing)* 3.11

`edible` *(thing)* 3.22, *9.1*

`enterable` *(container)* 3.6, *9.2*

`female` *(person)* 3.17

`fixed in place` *(thing)* 3.6, 3.7, 8.7

`handled` *(thing)* 3.11

`improper-named` *(thing)* 3.18

`indefinite article of` *(object)* 3.18

`inedible` *(thing)* 3.22, *9.1*

`initial appearance of` *(thing)* 3.11, 8.5

`lighted` *(room)* 3.15, *3.7*

`lit` *(thing)* 3.15, *3.7*

`lockable` *(container/door)* 3.13, *9.2*

`locked` *(container/door)* 3.13, *9.2*

`male` *(person)* 3.17

`marked for listing` *(thing)* 18.25, *3.1*

`matching key of` *(container/door)* 3.13

`mentioned` *(thing)* 18.24, 18.26, 18.28, *3.1*

`neuter` *(person)* 3.17

`non-recurring` *(scene)* 10.4, 10.7

`opaque` *(container)* 3.6, 3.15, *9.2*

`open` *(container/door)* 3.6, 3.13, 3.15, *9.2*

`openable` *(container)* 3.6, *9.2*

`opposite of` *(direction)* 3.26

`plural-named` *(thing)* 3.18

portable *(thing)* 3.6, 3.7, 8.7

printed name of *(object)* 8.5, 8.6, 18.10

printed plural name of *(object)* 18.11

proper-named *(thing)* 3.18

pushable between rooms *(thing)* 3.16

recurring *(scene)* 10.4, 10.7

scenery *(thing)* 3.8

singular-named *(thing)* 3.18

specification of *(thing)* 27.10

switched off *(device)* 3.14, *9.8*, *10.7*

switched on *(device)* 3.14, *9.8*, *10.7*

transparent *(container)* 3.6, 3.15, *9.2*

undescribed *(thing)* 3.24

unlit *(thing)* 3.15, *3.7*

unlocked *(container/door)* 3.13, *9.2*

unmarked for listing *(thing)* 18.25, *3.1*

unmentioned *(thing)* 18.24, 18.26, 18.28, *3.1*

unopenable *(container)* 3.6, *9.2*

wearable *(thing)* 3.20, *9.3*

changing multiple things with now 8.11

defining properties in Inform 6 27.21, 27.22

defining properties with a table 16.16, 16.17

defining sentence verbs for properties 15.12

dependence on kind 3.7

either/or properties 3.6, 4.6

changing 8.4

defining 4.7

used as adjectives 6.3

holding a new kind of value 4.9, 4.10

holding text 3.10, 4.6

implications 27.13

numeric comparatives, adjectives for 6.7, 15.12

numeric magnitudes, adjectives for 6.6

numeric superlatives, adjectives for 6.8, 15.12

of scenes 10.2, 10.4

shorthand name, omitting `of the` 8.6

understanding things by properties 17.15

value properties 3.10, 4.6

changing 8.5

defining 4.8, 4.9, 4.10

used as adjectives 6.3

proportional-spaced text 5.9, *12.1*

Proposal *example* 11.5 ex 171, *7.7 ex 171*

propriety 7.15

Provenance Unknown *example* 7.14 ex 107, *6.9 ex 107*

`provides`

provision relation *see* provision

provision *relation* 13.8

`public / private`, in release elements 25.16, 25.17

publication data *see* bibliographic data

`publicly-named / privately-named` *(object)* *adjective* 17.17, *2.2*

Puff of Orange Smoke *example* 12.10 ex 201, *7.5 ex 201*

Puncak Jaya *example* 17.16 ex 320, *5.5 ex 320*

`punctuated word number` *(number)* `in` *(text)* *phrase* 20.3

punctuation 2.3

apostrophe, meaning quotation mark 2.3, 5.2, 25.2

brackets

for calling values 8.15

for defining mistakes 17.21

for implicit multiplication 15.18

for phrase parameters 11.3

in parser clarifications 18.30

in response names 14.10

colon 6.4, 11.2, 11.7, 11.16, 19.3

comma

comma replacing colon 7.2

displaying serial comma 2.12, 5.5

in list constants, requiring spaces 21.2

separating phrase options 11.14

curly braces

for constant lists 21.2

for including Inform 7 expressions in Inform 6 27.15

double-dash

optional words in phrase definitions 11.3

optional words in Understand grammar 17.12

exclamation mark, ending sentences 2.3, 5.2

full stop

before text substitutions 5.6

ending sentences with line breaks 2.3, 5.2, 5.8

in counting words 20.3, 20.8

period *see* punctuation: full stop

question mark, ending sentences 2.3, 5.2

quotation marks

defining texts 2.3, 3.1, 5.1, 5.2

not usable in unit notations 15.14

removing from player's command 18.33, *6.13*, *7.1*, *7.7*

slash

separating synonymous words in grammar 17.4, 17.12, 17.13

separating synonymous words in phrase definitions 11.3

separating synonymous words in topics 16.13

square brackets

comments 2.3

text substitutions 2.3, 5.1, 5.2

vertical stroke, for paragraph break 2.3

Punctuation Removal *extension* *6.13*, *7.1*, *7.7*

PURLOIN *testing command* 2.8, 24.4

`pushable between rooms` *(thing)* *property* 3.16

pushing things 3.16

rules for pushing things 7.14, *6.9*

`"[put]"` *say phrase* 14.3

`putting it on` *action* *6.8*

Q

quantifiers

existential 6.2

universal 6.15

The Queen of Sheba *example* 16.13 ex 279, *7.11 ex 279*

Query *example* 17.21 ex 329, *11.3 ex 329*

querying the player yes/no 11.5

question mark *see* punctuation: question mark, ending sentences

Questionable Revolutions *example* 16.13 ex 278, *7.11 ex 278*

QUIT *11.6*

Quiz Show *example* 17.17 ex 323, *2.2 ex 323*

quotation marks *see* punctuation: quotation marks

quotations, displaying in boxes 5.12

R

Radio Daze *example* 5.7 ex 66, *9.9 ex 66*

radios (implementing) *9.9*

RANDOM *testing command* 24.3

`a/-- random` *(description of values)* *phrase* 8.19

`a random` *(name of kind)* `between` *(arithmetic value)* `and` *(arithmetic value)* *phrase* 8.18

`a random` *(name of kind)* `between` *(enumerated value)* `and` *(enumerated value)* *phrase* 8.18

`a random` *(name of kind)* `from` *(arithmetic value)* `to` *(arithmetic value)* *phrase* 8.18

`a random` *(name of kind)* `from` *(enumerated value)* `to` *(enumerated value)* *phrase* 8.18

randomness

- choosing a random table row 16.5
- combat with random results *7.5*
- dice and cards *9.5*
- enumerated values 8.18
- random numbers 8.18
- random things 8.19

randomizing the initial state of the world *11.1*

seeding the generator 8.18, 24.3

sorting lists in random order 21.8

sorting tables in random order 16.11

text variations 5.7, *2.1*

value kinds 8.18

rankings *see* scoring

Raszewski, L. Ross 1.2

`rather than`, in defining adjectives 6.4

rBGH *example* 15.8 ex 254, *5.1 ex 254*

`"['re]"` *say phrase* 14.8

`reaching inside / outside` *rulebook* 12.16, 12.18

`read` *(external file)* `into` *(table name)* *phrase* 23.13

reading *see* examining

`reading a command` *activity* 18.33

Real Adventurers Need No Help *example* 4.12 ex 54, *11.3 ex 54*

real numbers 15.2

arithmetic 15.5

conversion to/from numbers 15.3

displaying 15.4

exponential functions 15.6

given by equations 15.18

infinite and nonexistent 15.3

limits of numeric values 15.2

limits of numeric values of units 15.10, 15.14

trigonometric functions 15.7

units of measure *see* units of measure

`real square root of` *(arithmetic value)* *phrase* 15.5

Real Time Delays *extension* *12.4*

`reason the action failed` — *rule* *global variable* 12.5, 12.12, 19.14

`reciprocal of` *(real number)* *phrase* 15.6

reciprocal relations 13.6

recording devices (implementing) *9.12*

`recurring / non-recurring` *(scene)* *property* 10.4, 10.7

Red Cross *example* 12.7 ex 191, *7.5 ex 191*

(phrase (value, value) -> value) `reduction of` *(list of values)* *phrase* 22.5

`referring to`, in `understand as` 17.15

Reflections *example* 18.24 ex 352, *3.7 ex 352*

`refusal to act -- printing a refusal to act in the dark` *activity* 18.18

`"[regarding` *(number)*`]"` *say phrase* 14.5

`"[regarding` *(thing)*`]"` *say phrase* 14.5, 14.6

regional-containment *relation* 6.11

regions *kind* 3.4, 4.1

backdrops in regions 3.9

moving a backdrop to a new region 8.8

regional containment 6.11

rules applying to regions 7.11

things in regions 6.11

regular expressions 18.33, 20.6, 20.8, 20.9

syntax reference 20.9

`reject the player's command` *phrase* 18.33

`related by`, for understanding things by relations 17.16

(relation) `relates...` *assertion* 13.5, 13.12, 13.13

`relates`, for arbitrary relation conditions 13.14, 13.15

`relates`, for finding related values 13.13

relations 4.6, 13.3

creating things in relation to 4.15, 4.16

defining 13.5, 13.13

defining prepositions for relations 13.10

defining sentence verbs for relations 13.9

difference from sentence verbs 13.3

distance through 13.11

`empty / non-empty` *adjective* 13.14

expressing conditions 13.12

group relations 13.7

guidelines for use 13.16

indirect 13.11

involving values 13.13

kinds of relation 13.14

paths through 13.11

predefined

adjacency 6.14, 13.8

carrying 3.20, 8.7, 8.13, 13.4; *see also* carrying

concealment 3.24, 13.4, 18.9; *see also* concealment

containment 6.11, 8.13, 13.4; *see also* containment; containers

enclosure 3.25

equality 13.8

incorporation 3.23, 4.15, 8.13, 13.4; *see also* incorporation; components

mapping *(direction)* 6.14

meaning 14.9

possession 8.13, 13.4

regional-containment 6.11

support 8.13, 13.4; *see also* supporters

touchability 6.13; *see also* touchability

visibility 6.13; *see also* visibility

wearing 3.20, 8.7, 13.4

reciprocal relations 13.6

relationship to sentence verbs 13.8

temporary relations 13.15

understanding things by relations 17.16

used as values 13.14

used in sentences 13.1

RELATIONS *testing command* 13.7, 24.3

`release along with...` *assertion*

`a file of` *(description)* `called` *(filename)* 25.7

`a solution` 25.16

`a website` 25.10, 26.5

`an existing story file` 25.15

`an interpreter` 25.11, 25.12, 26.5

`an introductory booklet` 25.9

`an introductory postcard` 25.9

`cover art` 25.8

`public / private` elements 25.16, 25.17

`the library card` 25.17

`the source text` 25.17

Release button *user interface* 2.7, 25.6

release number *bibliographic data* 18.37, 25.2

release version (of the story) 2.9, 25.6, *13.1*

Relevant Relations *example* 14.9 ex 251, *2.1 ex 251*

`remainder after dividing` *(arithmetic value)* `by` *(arithmetic value)* *phrase* 15.5

REMEMBER *5.4*

Removal *example* 12.10 ex 198, *6.8 ex 198*

`remove entries` *(number)* `to` *(number)* `from` *(list of values)* *phrase* 21.5

`remove entry` *(number)* `from` *(list of values)* *phrase* 21.5

`remove` *(list of values)* `from` *(list of values)* *phrase* 21.5

`remove` *(object)* `from play` *phrase* 8.10

`remove` *(value)* `from` *(list of values)* *phrase* 21.5

`repeat through` *(table name)* *phrase* 16.6

`repeat through` *(table name)* `in reverse order` *phrase* 16.6

`repeat through` *(table name)* `in reverse` *(table column)* `order` *phrase* 16.6

`repeat through` *(table name)* `in` *(table column)* `order` *phrase* 16.6

`repeat with` *(a name not so far used)* `running from` *(arithmetic value)* `to` *(arithmetic value)* *phrase* 11.10

`repeat with` *(a name not so far used)* `running from` *(enumerated value)* `to` *(enumerated value)* *phrase* 11.10

`repeat with` *(a name not so far used)* `running through` *(description of values)* *phrase* 11.11

`repeat with` *(a name not so far used)* `running through` *(list of values)* *phrase* 21.4

`replace character number` *(number)* `in` *(text)* `with` *(text)* *phrase* 20.8

`replace line number` *(number)* `in` *(text)* `with` *(text)* *phrase* 20.8

`replace paragraph number` *(number)* `in` *(text)* `with` *(text) phrase* 20.8

`replace punctuated word number` *(number)* `in` *(text)* `with` *(text) phrase* 20.8

`replace` *(snippet)* `with` *(text) phrase* 18.33

`replace the punctuated word` *(text)* `in` *(text)* `with` *(text) phrase* 20.8

`replace the regular expression` *(text)* `in` *(text)* `with` *(text) phrase* 20.8

`replace the text` *(text)* `in` *(text)* `with` *(text) phrase* 20.8

`replace the word` *(text)* `in` *(text)* `with` *(text) phrase* 20.8

`replace unpunctuated word number` *(number)* `in` *(text)* `with` *(text) phrase* 20.8

`replace word number` *(number)* `in` *(text)* `with` *(text) phrase* 20.8

Replanting *example* 3.8 ex 16, *6.8 ex 16*

Replay button *user interface* 1.5, 1.7

`report` *(action) rulebook* 12.2, 12.9, *6.3*

- for other characters 12.13
- guidelines for use 12.21

`requiring`, in defining actions 12.7, 12.17, 17.1, *6.2*

resource IDs 23.10

RESPONSES *testing command* 14.12, 24.3

responses (library messages) 14.10, *6.16*

- changing 14.11
- `issuing the response text of something` *activity* 18.15
- listing during development 14.12, 24.3

Responsive *example* 14.11 ex 252, *2.1 ex 252*

RESTART *11.6*

RESTORE *11.6*

Results panel *user interface* 1.6

`resume the story` *phrase* 9.4

Revenge of the Fussy Table *example* 6.15 ex 82, *7.3 ex 82*

`reverse` *(list of values)* *phrase* 21.8

`reverse order`, for table rows 16.6, 16.11

reversed nouns, in understanding actions 17.2

Rideable Vehicles *extension* 3.16, *8.3*

`right hand status line` — *text* *global variable* 8.3, 18.23, *12.2*

Rip Van Winkle *example* 18.25 ex 355, *3.1 ex 355*

Roberts, Mike 1.2

Robo 1 *example* 21.5 ex 426, *7.12 ex 426*

Robo 2 *example* 21.9 ex 429, *7.12 ex 429*

Rock Garden *example* 18.29 ex 366, *3.4 ex 366*

Rocket Man *example* 20.4 ex 413, *2.1 ex 413*

`room-describing action` — *action name* *action variable* *6.9*

`room-or-door` *(direction)* `from/of` *(room)* *phrase* 6.14

`room description details -- printing room description details of something` *activity* 18.16, *3.1*

`room` *(direction)* `from/of` *(room)* *phrase* 6.14

`room gone from/to` — *object* *action variable* 19.16

rooms *kind* 3.5, 4.1, *3.2*

adjacent 6.14, 13.8

changing connections between rooms 8.5

connections between rooms 3.2, *3.2*

creating 3.1

descriptions 2.12, *3.1*, *6.4*

moving without printing one 8.7, 8.9

divided into smaller areas *3.3*

grouping into regions 3.4, 6.11

listening to (with no object) 7.7, *3.8*

moving rooms *8.2*

name of a dark room 18.21

one-way connections between rooms 3.3

printing the room contents 18.26, 18.27, 18.28, *3.1*

rules applying to rooms 7.11, 7.13

smelling (with no object) 7.7

things in more than one room 3.9, *3.4*, *3.9*

roots, of numbers 15.5

ropes (implementing) *10.6*

Rossing, Eric *1.2*

`rotate` *(list of values)* *phrase* 21.8

`rotate` *(list of values)* `backwards` *phrase* 21.8

route-finding, fast/slow 6.14, 13.11

rows, in tables *see* tables

`rows -- choose a/the/-- row` *(number)* `in/from` *(table name)* *phrase* 16.5

`rows -- choose a/the/-- row with` *(table column)* `of` *(value)* `in/from` *(table name)* *phrase* 16.5

`rows -- number of rows in/from` *(table name)* *phrase* 16.2

Rubies *example* 23.13 ex 442, *11.4 ex 442*

`rule fails` *phrase* 19.11

`rule succeeds` *phrase* 19.11

`rule succeeds with result` *(value)* *phrase* 19.13

rulebooks 19.1, 19.2

- as a kind of rule 19.8
- basis of rulebook 19.9
- defining 19.8
- `empty / non-empty` *adjective* 19.8
- for actions 12.2, *6.3*
- for actitivies 18.3, 18.5
- for new actions 12.9, *6.2*
- invoking a rulebook 19.8
  - and abiding by its outcome 19.14
- multiple action processing rules 17.20

order of rules 19.2, 19.4, 19.6, 19.16

outcomes 19.11, 19.12, 19.13

rulebook parameters 19.9

stopping execution 19.11

variables for rulebooks 19.10

rules 7.2

abbreviating actions with `doing it` 7.17

access to things referred to in actions 12.16, 12.18

actions referred to in conditions 9.12

after rules 7.5, 12.2, 12.15, 18.3, *6.3*

applying only when characters are present 7.12

applying only while performing activities 18.4

applying to all but some actions 7.9

applying to arbitrary things 7.10

applying to consecutively repeated actions 7.17

applying to multiple actions 7.8

applying to repeated actions 7.16

applying to rooms and regions 7.11, 7.13

arbitrary conditions for rules, using `when` 7.12, 9.5

arbitrary conditions for rules, using `while` 18.4

before rules 7.3, 12.2, 12.15, 18.3, *6.3*

changing conditions of rules 19.5

counting repetitions for conditions 9.14

counting turns for conditions 9.15

creating new kinds of action to reduce redundancy 7.15

defining 2.2, 19.3, 19.7

defining in Inform 6 27.20

failure of rules 19.11

for actions found in rulebooks 12.2, *6.3*

for activities 18.3, 18.5

for any actor (including the player) 12.14

for `exiting` 7.14

for `going`, special cases 7.13, 7.14

for new actions 12.9, *6.2*

for new actions for other characters 12.11

for other characters 12.4, 12.5, 12.11, 12.12, 12.13

for `pushing` objects to other rooms 7.14

for scenes 10.2, *4.4*

for senses 7.7

for speech 7.6

for visibility 12.19

guidelines for action rules 12.21

in rulebooks 19.2

instead rules 7.2, 7.3, 12.2, 12.4, 12.15, *6.3*

invoking a rule 19.8

and abiding by its outcome 19.14

naming 19.3, 19.7

order in rulebooks 19.2, 19.4, 19.6, 19.16

past tense of conditions 9.12, 9.13

perfect tense of conditions 9.13

persuasion rules for instructing characters 12.4

preamble 19.7

reaching inside / outside 12.16, 12.18

removing 19.4, 19.5, *6.3*

replacing 19.4, 19.5, *6.3*

run at beginning of scene 10.2, *4.4*

run at beginning of story 9.1, 19.15, *11.1*

run at end of scene 10.2, *4.4*

run at end of story 9.4, 19.15, *11.6*

run every turn 9.5, *7.2*

success of rules 19.11, 19.13

unsuccessful attempts by other characters 12.5

RULES *testing command* 24.3, *6.2*

Rules of Attraction *example* 18.16 ex 346, *10.7 ex 346*

Rules page of Index panel *user interface* 19.6, 19.16

`running from`, looping through values 11.10

`running through`, looping through objects 11.11

S

sack object *see* hold-all

Safari Guide *example* 17.10 ex 307, *6.9 ex 307*

Safety *example* 17.6 ex 298, *9.2 ex 298*

Saint Eligius *example* 19.4 ex 393, *6.9 ex 393*

`same` *(property)* `as` 6.7

Sand *example* 7.3 ex 87, *6.8 ex 87*

Savannah *example* 15.19 ex 267, *10.2 ex 267*

`say "[']"` *phrase* 5.2

`say "['re]"` *phrase* 14.8

`say "['ve]"` *phrase* 14.8

`say "[a list of` *(description of objects)*`]"` *phrase* 5.5

`say "[A list of` *(description of objects)*`]"` *phrase* 5.5

`say "[a list of` *(description of objects)* `including contents]"` *phrase* 5.5

`say "[A` *(object)*`]"` *phrase* 5.3

`say "[a` *(object)*`]"` *phrase* 5.3

`say "[adapt` *(verb)*`]"` *phrase* 14.9

`say "[adapt` *(verb)* `from` *(narrative viewpoint)*`]"` *phrase* 14.9

`say "[adapt` *(verb)* `in` *(grammatical tense)*`]"` *phrase* 14.9

`say "[adapt` *(verb)* `in` *(grammatical tense)* `from` *(narrative viewpoint)*`]"` *phrase* 14.9

say "[an *(object)*]" *phrase* 5.3

say "[An *(object)*]" *phrase* 5.3

say "[apostrophe/']" *phrase* 5.2

say "[are]" *phrase* 14.2

say "[aren't]" *phrase* 14.8

say "[as decreasingly likely outcomes]" *phrase* 5.7

say "[at random]" *phrase* 5.7

say "[bold type]" *phrase* 5.9, *12.1*

say "[bracket]" *phrase* 5.2

say "[can]" *phrase* 14.2, 14.7

say "[can't]" *phrase* 14.7

say "[can see]" *phrase* 14.2

say "[can touch]" *phrase* 14.2

say "[carry]" *phrase* 14.2

say "[close bracket]" *phrase* 5.2

say "[*(column name)* in/from table *(table name)*]" *phrase* 16.4

say "[command clarification break]" *phrase* 5.8

say "[conditional paragraph break]" *phrase* 5.8

say "[could]" *phrase* 14.7

say "[couldn't]" *phrase* 14.7

say "[current table row]" *phrase* 16.4

say "[cycling]" *phrase* 5.7

say "[don't]" *phrase* 14.8

say "[else]" *phrase* 5.6

say "[end if]" *phrase* 5.6

say "[end unless]" *phrase* 5.6

say "[first time]" *phrase* 5.7

say "[fixed letter spacing]" *phrase* 5.9, *12.1*

say "[have]" *phrase* 14.2

say "[haven't]" *phrase* 14.8

say "[here]" *phrase* 14.1

say "[if *(a condition)*]" *phrase* 5.6

say "[in random order]" *phrase* 5.7

say "[infinitive of *(verb)*]" *phrase* 14.9

say "[is-are a list of *(description of objects)*]" *phrase* 5.5

say "[is-are list of *(description of objects)*]" *phrase* 5.5

say "[is-are the list of *(description of objects)*]" *phrase* 5.5

say "[it]" *phrase* 14.5

say "[italic type]" *phrase* 5.9, *12.1*

say "[line break]" *phrase* 5.8

say "[list of *(description)*]" *phrase* 6.1

say "[list of *(description of objects)*]" *phrase* 5.5

say "[*(list of objects)* with definite articles]" *phrase* 21.3

say "[*(list of objects)* with indefinite articles]" *phrase* 21.3

say "[*(list of values)* in brace notation]" *phrase* 21.3

say "[may]" *phrase* 14.7

say "[mayn't]" *phrase* 14.7

say "[might]" *phrase* 14.7

say "[mightn't]" *phrase* 14.7

say "[must]" *phrase* 14.7

say "[mustn't]" *phrase* 14.7

say "[negate *(verb)*]" *phrase* 14.9

say "[negate *(verb)* from *(narrative viewpoint)*]" *phrase* 14.9

say "[negate *(verb)* in *(grammatical tense)*]" *phrase* 14.9

say "[negate *(verb)* in *(grammatical tense)* from *(narrative viewpoint)*]" *phrase* 14.9

say "[no line break]" *phrase* 5.8

say "[now]" *phrase* 14.1

say "[*(number)* in words]" *phrase* 5.4

say "[one of]" *phrase* 5.7

say "[only]" *phrase* 5.7

say "[or]" *phrase* 5.7

say "[otherwise]" *phrase* 5.6

say "[otherwise/else if *(a condition)*]" *phrase* 5.6

say "[otherwise/else unless *(a condition)*]" *phrase* 5.6

say "[our]" *phrase* 14.4

say "[paragraph break]" *phrase* 5.8

say "[past participle of *(verb)*]" *phrase* 14.9

say "[possessive]" *phrase* 14.6

say "[present participle of *(verb)*]" *phrase* 14.9

say "[purely at random]" *phrase* 5.7

say "[put]" *phrase* 14.3

say "[quotation mark]" *phrase* 5.2

say "[*(real number)* in decimal notation]" *phrase* 15.4

say "[*(real number)* in scientific notation]" *phrase* 15.4

say "[*(real number)* to *(number)* decimal places]" *phrase* 15.4

say "[*(real number)* to *(number)* decimal places in decimal notation]" *phrase* 15.4

say "[*(real number)* to *(number)* decimal places in scientific notation]" *phrase* 15.4

say "[regarding *(number)*]" *phrase* 14.5

say "[regarding *(thing)*]" *phrase* 14.5, 14.6

say "[roman type]" *phrase* 5.9, *12.1*

say "[row *(number)* in/from table *(table name)*]" *phrase* 16.4

say "[run paragraph on]" *phrase* 5.8

say "[run paragraph on with special look spacing]" *phrase* 5.8

say "[s]" *phrase* 5.4

say "[*(sayable value)*]" *phrase* 5.3

say "[should]" *phrase* 14.7

say "[shouldn't]" *phrase* 14.7

say "[sticky random]" *phrase* 5.7

say "[stopping]" *phrase* 5.7

say "[text of *(external file)*]" *phrase* 23.14

say "[text of *(response)*]" *phrase* 14.10

say "[the/-- banner text]" *phrase* 18.37

say "[the/-- complete list of extension credits]" *phrase* 27.6

say "[the/-- list of extension credits]" *phrase* 27.6

say "[the/-- player's surroundings]" *phrase* 8.3

say "[the list of *(description of objects)*]" *phrase* 5.5

say "[The list of *(description of objects)*]" *phrase* 5.5

say "[The *(object)*]" *phrase* 5.3

say "[the *(object)*]" *phrase* 5.3

say "[their]" *phrase* 14.5

say "[then at random]" *phrase* 5.7

say "[then purely at random]" *phrase* 5.7

say "[there]" *phrase* 14.5

say "[they]" *phrase* 14.2, 14.5

say "[those]" *phrase* 14.6

say "[*(time)* in words]" *phrase* 9.7

say "[unless *(a condition)*]" *phrase* 5.6

`say "[us]" phrase` 14.4

`say "[variable letter spacing]" phrase` 5.9, *12.1*

`say "[we]" phrase` 14.4

`say "[wear]" phrase` 14.2

`say "[won't]" phrase` 14.8

`say "[would]" phrase` 14.7

`say "[wouldn't]" phrase` 14.7

`sayable value` 22.1, 22.6

`scaled at` 15.10

`scaled up / down by` 15.9, 15.10

`scenery` *(thing)* *property* 3.8

backdrops made automatically scenery 3.9

scenes 10.1

beginning when play begins 10.6

conditions for scenes 10.4

descriptions (displayed when scene begins) 10.2

`Entire Game` 10.3

for the passage of story time *4.1*

linking to other scenes 10.5, 10.6, 10.7, 10.8

no way to force scenes to begin / end 10.9

properties of scenes 10.2, 10.4

recurring scenes 10.4, 10.7

rules run at beginning 10.2, *4.4*

rules run at end 10.2, *4.4*

scripting story events in scenes *4.2*

times of scenes beginning/ending 10.3

variations in ending 10.8

SCENES *testing command* 10.3, 24.3, *13.1*

Scenes page of Index panel *user interface* 10.3, 10.9

scientific notation 15.2, 15.4

scope 17.7, 18.29, *3.6*, *5.5*

SCOPE *testing command* 24.4

`score` — *number global variable* 8.3, 9.2, *11.4*

scoring 9.2, *11.4*

enabling 2.12, 9.2

in status line 8.3

maximum score 9.2

ranking table 9.3

reporting at end of story *11.6*

reporting during story 9.2, *11.4*, *11.5*

`scoring` *use option* 2.12, 9.2, *11.4*

screen top 8.3, 18.23, *12.2*

SCRIPT ON/OFF *see* TRANSCRIPT ON/OFF

Search and Seizure *example* 3.24 ex 38, *7.3 ex 38*

`searching` *action* *6.6*

`second noun part of` *(action)* *phrase* 12.20

`second noun` — *object* *global variable* 7.10

The Second Oldest Problem *example* 12.10 ex 200, *6.9 ex 200*

`second person singular / plural` 14.1

`section` *of source text* 2.5

`see` *relation verb* *see* visibility

`seed the random-number generator with` *(number)* *phrase* 8.18

`seldom / usually`, in assertions 4.3

senses 7.7, *3.8*

`sentence case --` *(text)* `in sentence case` *phrase* 20.4

sentence verbs 13.1, 13.2

- as values 14.9
- defining for properties 15.12
- defining for relations 13.9
- defining for relations with prepositions 13.10
- difference from relations 13.3
- predefined 13.8

`serial comma` *use option* 2.12, 2.13, 5.5, 21.3

`set pronouns from` *(object)* *phrase* 17.18

`set the locale priority of` *(object)* `to` *(number)* *phrase* 18.27

`setting action variables` *rulebook* 12.10

Shawn's Bad Day *example* 17.4 ex 294, *6.15 ex 294*

Shipping Trunk *example* 18.10 ex 339, *9.2 ex 339*

Short, Emily 1.2, 2.10, 18.37, 25.9, 27.4, 27.7, *3.5*, *4.5*, *6.13*

shorthand *see* abbreviations

`"[should]"` *say phrase* 14.7

`"[shouldn't]"` *say phrase* 14.7

`show relation` *(relation of values to values)* *phrase* 13.7

SHOWHEAP *testing command* 24.4

SHOWME *testing command* 2.7, 24.2, 24.3, 24.4, *13.1*

`showme the contents of` *(table name)* *phrase* 16.4

`showme` *(value)* *phrase* 11.4

SHOWVERB *testing command* 24.4

siblings, in containment hierarchy 8.17

Sieve of Eratosthenes *example* 21.11 ex 437, *10.11 ex 437*

Signs and Portents *example* 4.9 ex 53, *9.8 ex 53*

`silently try` *(action)* *phrase* 7.4, 12.1

`sine of` *(real number)* *phrase* 15.7

`singular-named / plural-named` *(thing)* *property* 3.18

Situation Room *example* 9.7 ex 142, *4.1 ex 142*

Skein panel *user interface* 1.7, 1.8, 1.9, 25.16, *13.1*

Skeleton Keys *extension* *9.2*

slash *see* punctuation: slash

`sleeping` *action* *6.11*

Slightly Wrong *example* 3.1 ex 4, *3.1 ex 4*

Slogar's Revenge *example* 12.17 ex 214, *6.3 ex 214*

Slouching *example* 19.4 ex 396, *5.1 ex 396*

`slow route-finding` *use option* 6.14

for particular relations 13.11

`smelling` *action* 7.7

smoke (implementing) *10.1*

Snip *example* 15.16 ex 261, *10.6 ex 261*

snippets 17.5, 18.33

Solitude *example* 19.8 ex 400, *11.3 ex 400*

`some`

article 3.18

in descriptions 6.15

Some Assembly Required *example* 17.22 ex 332, *9.3 ex 332*

`somebody`, in descriptions 6.2

`someone`

in actions 12.12, 12.13

in descriptions 6.2

`[someone]` *grammar token* 17.1, 17.4

`something`

in descriptions 6.2

naming value of 8.15

`[something]` *grammar token* 17.1, 17.2, 17.4

Something Narsty *example* 4.3 ex 45, *3.5 ex 45*

`something new`, in understanding actions 17.3

`[something preferably held]` *grammar token* 17.4

`[something related by` *(relation)*`]` *grammar token* 17.16

`somewhere`, in descriptions 6.2

`sort` *(list of objects)* `in` *(property)* `order` *phrase* 21.8

`sort` *(list of objects)* `in reverse` *(property)* `order` *phrase* 21.8

`sort` *(list of values)* *phrase* 21.8

`sort` *(list of values)* `in random order` *phrase* 21.8

`sort` *(list of values)* `in reverse order` *phrase* 21.8

`sort` *(table name)* `in random order` *phrase* 16.11

`sort` *(table name)* `in reverse` *(table column)* `order` *phrase* 16.11

`sort` *(table name)* `in` *(table column)* `order` *phrase* 16.11

`sound name` 23.10

sounds 23.7, *12.5*

accessibility text 23.9

as values 23.10

declaring 23.8

formats allowed 23.7

placing in materials folder 23.8

source text

releasing with the story 25.17

subdivisions 2.5, 2.6, 2.9, 25.6

Space Patrol - Stranded on Jupiter! *example* 10.4 ex 158, *4.4 ex 158*

spaces, in list constants, after commas 21.2

spaces vs tabs (in tables) 9.3, 16.1

`specification of` *(thing)* *property* 27.10

(spec) `specifies` *(a unit)* *assertion* 15.8

The Speed of Thought *example* 15.20 ex 270, *10.5 ex 270*

Spellbreaker *example* 12.15 ex 210, *11.2 ex 210*

Spencer, Henry 20.6

Spring Cleaning *example* 8.10 ex 123, *10.4 ex 123*

square brackets *see* punctuation: square brackets

`square root of` *(arithmetic value)* *phrase* 15.5

Standard Rules *extension* 27.2, 27.14

standing *see* exiting

Starry Void *example* 3.2 ex 7, *3.3 ex 7*

starting conditions 2.1, 3.1

scenes that begin at the beginning 10.6

setting in `when play begins` 9.1, *11.1*

`starting the virtual machine` *activity* 18.40

Stately Gardens *example* 18.29 ex 367, *3.4 ex 367*

status line 8.3, 18.23, *12.2*

Status line with centered text, the hard way *example* 27.19 ex 452, *12.2 ex 452*

Stone *example* 19.3 ex 391, *9.1 ex 391*

`stop` *phrase* 11.13

`stop the action` *phrase* 7.3, 19.11

using `instead` *see* (instead), to stop the action

stored actions 12.20

story description *bibliographic data* 25.2, 26.3

story structure

beginning 9.1, *11.1*; *see also* initial state of the world

changing viewpoint *5.6*

characterization *5.3*, *7.10*

cut scenes *4.5*, *7.12*

dialogue *7.6*, *7.9*

dynamic plot management *4.6*

ending 9.4, *11.6*; *see also* end of story

flashbacks *4.5*

providing background information *5.4*

scenes in different environments *4.4*

scenes with scripted events *4.2*

`story tense` — *tense global variable* 14.1

`story viewpoint` — *narrative viewpoint global variable* 14.1

Straw Boater *example* 4.8 ex 49, *8.1 ex 49*

Straw Into Gold *example* 17.9 ex 305, *2.1 ex 305*

Strictly Ballroom *example* 11.11 ex 176, *7.16 ex 176*

string *see* ropes (implementing)

strings of characters *see* text

subheadings 2.5, 2.6, 2.9, 25.6

`substituted / unsubstituted` *(text) adjective* 20.7

`substituted form of` *(text) phrase* 20.7

success

for actions 7.1, 12.2

for persuasion 12.4

for rules 19.11, 19.13

`sufficient / insufficient light, there is sufficient / insufficient light` *(visibility rules) outcome phrase* 12.19

`SUPERBRIEF room descriptions` *use option* 2.12, *6.4*, *11.5*

superlatives

of numbers using adjectives 6.8

of numbers with units using adjectives 15.12

`supplying a missing noun/second noun` *activity* 17.1, 18.32

support *relation* 8.13, 13.4

`supporter in question` — *object global variable* 12.18

supporters *kind* 3.5, *8.4*

carrying capacity 3.19

default immobility 3.7

mentioned because of supported things 3.8

support relation *see* support

Sweeney *example* 16.16 ex 282, *7.8 ex 282*

Swerve left? Swerve right? Or think about it and die? *example* 13.10 ex 232, *10.5 ex 232*

Swigmore U. *example* 19.4 ex 397, *8.4 ex 397*

`switched on / off` *(device)* *property* 3.14, *9.8*, *10.7*

Sybil 1 *example* 7.6 ex 91, *7.7 ex 91*

Sybil 2 *example* 7.6 ex 93, *7.7 ex 93*

symbols *see* characters (letters)

`symmetric` *(relation)* *adjective* 13.14

synonyms 12.7, *6.2*

T

`table name` 16.15

`Table of Final Question Options` *11.6*

tables 9.3, 16.1

adding rows 16.10

amending tables 16.19

blank columns 16.8, 16.10

blank entries 16.7, 16.10

blank rows 16.9, 16.10

changing entries 16.4

choosing rows 16.5

continuing tables 16.18

creating multiple things with a table 16.16

defining 16.1

defining a kind of value with a table 16.17

defining multiple kinds with a table 16.16

describing things listed in tables 16.12

`empty / non-empty` *adjective* 16.9

formatting 16.1

looking up by cross-reference 16.3

looking up by row number 16.2

ranking score 9.3

reading / writing to data files 23.13

removing rows 16.10

repeating through 16.6

sorting tables 16.11

tabs in formatting tables 9.3, 16.1

topic columns 16.13

used as values 16.15

tabs, for indentation *of source text* 11.7, 11.9

tabs vs spaces, in tables *of source text* 9.3, 16.1

`taking` *action* 7.1, *6.8*

implicit taking 12.17, 18.34, *6.8*

`taking inventory` *action* *6.7*

Tamed *example* 3.6 ex 13, *8.4 ex 13*

`tangent of` *(real number)* *phrase* 15.7

Tarbert, Mike *1.2*

`tasting` *action* 7.7

Technological Terror *example* 8.11 ex 127, *7.5 ex 127*

`telemetry recordings` *use option* 2.13

telephones (implementing) *9.10*

televisions (implementing) *9.9*

`telling` *(actor)* `about` *action* 7.6

templates, for web pages for the story 25.13, 25.14

templates, Inform 6 template layer 27.26, *6.3*

Temple, Erik 1.2, *12.4*

temporary variables 11.15

displaying 20.7

tense

of conditions in rules 9.12, 9.13

of standard responses 14.1, *5.6*

Tense Boxing *example* 9.13 ex 149, *5.5 ex 149*

Terracottissima *example* 17.15 ex 314, *10.4 ex 314*

Terracottissima Maxima *example* 17.15 ex 317, *2.3 ex 317*

Terror of the Sierra Madre *example* 8.9 ex 121, *5.6 ex 121*

Tessman, Kent 1.2

TEST *testing command* 2.8, 24.2

`test` *(test name)* `with` *(commands)* *assertion* 2.8, 24.2

test version (of the story) 2.9, 24.5, *13.1*

testing commands 2.9, *13.1*

ABSTRACT 24.4

ACTIONS 7.1, 12.1, 12.5, 12.12, 24.3, *10.10*, *13.1*

checking story correctness in the Index 24.1

comments in transcripts from beta testers 17.21, 24.7, *13.1*

defining 2.9, 24.5, *13.1*

GONEAR 24.4

PURLOIN 2.8, 24.4

RANDOM 24.3

RELATIONS 13.7, 24.3

RESPONSES 14.12, 24.3

RULES 24.3, *6.2*

SCENES 10.3, 24.3, *13.1*

SCOPE 24.4

SHOWHEAP 24.4

SHOWME 2.7, 24.2, 24.3, 24.4, *13.1*

SHOWVERB 24.4

TEST 2.8, 24.2

TRACE 24.4

TREE 24.4

text 5.1, 20.1

accessibility text for figures and sounds 23.9

changing case 20.4

counting characters / words / paragraphs / lines 20.3

displaying 5.1

`empty / non-empty` *adjective* 6.5, 20.3

in properties 3.10

indexed text, now the same as text 20.1

lower case 20.4

matching 20.5

memory limits 20.2

reading / writing / appending to data files 23.14

recording from the player's command *2.3*

regular expressions 18.33, 20.6, 20.8, 20.9

replacing parts of a text 20.8

`substituted / unsubstituted` *adjective* 20.7

substitutions *see* text substitutions

upper case 20.4

when text substitutions are substituted 20.7

Z-machine limitations 20.2, 20.4

`[text]` *grammar token* 17.5

in understanding mistakes 17.21

text files *see* files (data files)

Text Foosball *example* 9.5 ex 140, *7.2 ex 140*

`text matching regular expression` *phrase* 20.6

`text matching subexpression` *(number)* *phrase* 20.6

text substitutions 3.14, *2.1*

defining 5.13

enumerated values 4.9

lists 5.5, 21.3

new kinds of value 4.9

numbers 5.3, 5.4

punctuation 5.2

random variations 5.7

real numbers 15.4

sayable values 22.1, 22.6

sequential variations 5.7

text 5.1

things 5.3

units of measure 15.8, 15.11, 15.15

values 5.3

variations 5.6

when text is substituted 20.7

(name of kind) `that/which/who relates to` *(value)* `by` *(relation of values to values)* *phrase* 13.13

(name of kind) `that/which/whom` *(value)* `relates to by` *(relation of values to values)* *phrase* 13.13

that, adaptive pronouns in displayed text 14.6

`that`, in description clauses 6.9

`that varies`, in defining global variables 4.12

`the / a / an` 3.18, 5.3, 5.5

`"[their]"` *say phrase* 14.5

THEM, in player commands 17.18, *2.2*

`"[there]"` *say phrase* 14.5

`there is`, declaring existence 6.10

`there is sufficient / insufficient light` *(visibility rules)* *outcome phrase* 12.19

"[they]" *say phrase* 14.2, 14.5

things *kind* 3.5, 4.1

changing location of 8.7, 8.8, 8.9

changing multiples with `now` 8.11

counting 8.16

creating 3.1

in Inform 6 27.21

multiple identical things with counts 4.14

multiple things with a table 16.16

using plurals 4.4

using relations 4.15, 4.16

displaying 5.3

immobile

because they're supporters 3.7

`fixed in place` 3.6, 8.7

moving anyway 8.7

`scenery` 3.8

in more than one room 3.9

kinds of 4.1, 4.2, 4.4

location of 3.25

parts of 4.15

parts of things 3.23

pushable between rooms 3.16, 7.14, *6.9*

references to (descriptions) 6.1

referring to arbitrary things in rules 7.10

removing from play 8.10

starting out of play 3.25, 6.10, 8.10

`[things]` *grammar token* 17.4

`[things inside]` *grammar token* 17.4

THINK ABOUT *5.4*

`third person singular / plural` 14.1

Thirst *example* 8.5 ex 117, *10.2 ex 117*

Thirst 2 *example* 8.5 ex 118, *10.8 ex 118*

`this is`, in naming phrases 22.3

`this is the` *(name)* `rule...` *assertion* 19.3, 19.7

Thompson, Ken 20.6

Thornton, Adam 1.2

`"[those]"` *say phrase* 14.6

3 AM *example* 12.7 ex 193, *10.2 ex 193*

`through` *relation verb* 3.12

Tilt 1 *example* 17.15 ex 318, *9.5 ex 318*

Tilt 2 *example* 19.13 ex 407, *9.5 ex 407*

Tilt 3 *example* 27.7 ex 450, *12.1 ex 450*

time *3.9*, *4.1*

calculations 9.9, 9.10

clocks (implementing) *9.11*

comparing 9.9

conditions and rules referring back in time 9.13

counting actions with `for the Nth time` 7.16, *11.4*

counting conditions with `for the Nth time` 9.14

displaying approximations 9.8

displaying times 9.7

events in the future 9.11

not passing for `out of world` actions 12.15

scene beginning/ending times 10.3

scripting story events by time *4.3*

time of day 9.6

timed input *12.4*

waiting intervals of time *6.11*

`[time]` *grammar token* 17.9

`time of day` — *time global variable* 8.2, 8.3, 9.6, 9.7

`[time period]` *grammar token* 17.9

`time since` *(scene)* `began` *phrase* 10.3

`time since` *(scene)* `ended` *phrase* 10.3

`time when` *(scene)* `began` *phrase* 10.3

`time when` *(scene)* `ended` *phrase* 10.3

Timeless *example* 19.15 ex 409, *4.1 ex 409*

(arithmetic value) `times` *(arithmetic value)* *phrase* 15.5

Tiny Garden *example* 5.5 ex 62, *3.4 ex 62*

title *bibliographic data* 2.5, 18.37, 25.2

`title case --` *(text)* `in title case` *phrase* 20.4

to be *see* is

`to decide what/which` *(value)* *assertion* 11.17

`to decide whether/if` *assertion* 11.16

`to` *(phrase name)`...`* *assertion* 11.2, 19.7

`to say` *(substitution)`...`* *assertion* 5.13

(arithmetic value) `to the nearest` *(arithmetic value)* *phrase* 9.8, 15.5

(real number) `to the nearest whole number` *phrase* 15.3

(real number) `to the power` *(real number)* *phrase* 15.6

`to` *(V)* `is a verb` *assertion* 14.3, 14.7

(name of kind) `to which/whom` *(value)* `relates by` *(relation of values to values)* *phrase* 13.13

Today Tomorrow *example* 7.12 ex 99, *8.3 ex 99*

`together --` `group` *(description of objects)* `together` *phrase* 18.13

tokens of grammar *see* grammar tokens

Tom's Midnight Garden *example* 17.6 ex 299, *9.11 ex 299*

`topic understood` — *snippet global variable* 17.5

topics

- in table columns 16.13
- understanding 17.5

`total` *(arithmetic values valued property)* `of` *(description of values) phrase* 15.17

Totality example 9.11 ex 144, *3.9 ex 144*

touchability *relation* 6.13

- in action processing sequence 12.2, 12.16
- requirement in defining actions 12.7, 12.17, 12.18

`touchable / untouchable` *(thing) adjective* 6.13

`touching` *action* 7.7

TRACE *testing command* 24.4

Trachypachidae Maturin 1803 example 18.10 ex 340, *9.2 ex 340*

TRANSCRIPT ON/OFF 24.7, *11.5*

Transcript panel *user interface* 1.8, 1.9

transcripts

- creating with the TEST command 2.8
- in the Skein 1.8, 1.9

`translates as...` *assertion*

- actions 27.22
- global variables 27.22
- kinds 27.21

properties 27.21, 27.22

rules 27.20

things 27.21

understanding (grammar tokens) 27.23

Unicode characters 27.25

use options 27.18

Transmutations *example* 13.7 ex 227, *10.10 ex 227*

`transparent / opaque` *(container)* *property* 3.6, 3.15, *9.2*

Treaty of Babel 25.5

TREE *testing command* 24.4

Trieste *example* 16.19 ex 285, *11.3 ex 285*

The Trouble with Printing *example* 17.3 ex 292, *9.6 ex 292*

`true / false` 11.5

`truncate` *(list of values)* `to` *(number)* `entries/entry` *phrase* 21.10

`truncate` *(list of values)* `to the first` *(number)* `entries/entry` *phrase* 21.10

`truncate` *(list of values)* `to the last` *(number)* `entries/entry` *phrase* 21.10

`truth state` 11.5

`try` *(action)* *phrase* 7.4, 12.1

in `asking` action *see* asking (actor) to try

`try` *(actor)* `trying` *(action)* *phrase* 12.6

`try silently` *(action)* *phrase* 7.4

Trying Taking Manhattan *example* 12.11 ex 204, *6.7 ex 204*

`turn count` — *number* *global variable* 5.4, 8.3, 9.5

`turn sequence` *rulebook* 19.15

turning devices on / off 3.14, *9.8*, *10.7*

turns

- counting actions with `for the Nth turn` 7.17
- counting conditions with `for the Nth turn` 9.15
- events in the future 9.11
- not passing for `out of world` actions 12.15
- not passing for specific actions *4.1*
- rules run every turn 9.5
- turn sequence 19.15
- using time of day instead of turns 9.6

`twice`, counting conditions 9.14

type-checking 8.1, 8.4

- for global variables 4.12
- for temporary variables 11.15
- in conditions 11.6
- in grammar tokens 17.14
- in input grammar 17.6
- in lists 21.1
- in phrase definitions 11.3

in phrases that decide values 11.17

in tables 16.1, 16.4

of relation types 13.14

of units of measure 15.8, 15.19, 15.20

types *see* kinds

U

U-Stor-It *example* 4.15 ex 58, *8.4 ex 58*

`unabbreviated object names` *use option* 3.2

Unblinking *example* 6.14 ex 81, *3.7 ex 81*

The Unbuttoned Elevator Affair *example* 3.3 ex 9, *8.2 ex 9*

Uncommon Ground *example* 27.30 ex 455, *5.6 ex 455*

Under Contract *example* 12.11 ex 205, *7.14 ex 205*

`understand nothing as` *(action)* *assertion* 17.3

`understand` *(nouns)* `as` *(thing)* *assertion* 17.8, 17.11, 17.15, 17.16

`understand` *(property)* `as describing/referring to` *(thing)* *assertion* 17.15

`understand the command` *(verb)* `as` *(verb)* *assertion* 17.2

`understand` *(verb)* `as` *(action)* *assertion* 12.7, 12.8, 17.1, 17.2, 17.3, *6.2*

`understand` *(verb)* `as something new` *assertion* 17.3

`understand` *(words)* `as a mistake` *assertion* 17.21, *5.3*

understanding 17.1, *2.2*

adding synonyms for verbs 17.2

alternatives to standard parsing *6.18*

arbitrary objects 17.4

arbitrary text 17.5

for conversations and consulting 7.6

choosing between multiple grammar lines 17.22

conditionally 17.17, *2.2*

deciding whether ALL includes something 18.36, *6.15*

disambiguation *see* disambiguation: of player commands

error messages for parsing errors 18.35, *6.17*

grammar tokens *see* grammar tokens

kinds of things 17.11

limiting cases where understand rules apply 17.17, *2.2*

mistakes 17.21, *5.3*

noun-first sentences 17.10

numbers 17.6, 17.9

numbers with units 15.16

precedence of commands understood 17.22

properties as describing things 17.15

recording the player's command *2.3*

relations as describing things 17.16

removing existing definitions of verbs 17.3, 17.3

requirements for objects in actions 12.7, 12.17, 12.18, *6.2*

sentences without verbs 17.10

special processing of keyboard input 18.33

synonyms 17.8, 17.12

as grammar tokens 17.13

for things and values 17.11

things

by properties 17.15

by relations 17.16

kinds of things 17.11

not in scope with `any` 17.7, *5.5*

specific things 17.8

values 17.11

grammar tokens producing values 17.14

kinds of value 17.9

numbers 17.6

verbs 12.7, 12.8, 17.1, 17.2, 17.3, *6.2*

`when` clauses for 17.17

with Inform 6 functions 27.23

(kind) `understood` *— kind of value global variable* 17.9

The Undertomb 1 *example* 4.9 ex 50, *3.8 ex 50*

The Undertomb 2 *example* 4.9 ex 51, *3.7 ex 51*

`undescribed / described` *(thing) property* 3.24

UNDO 2.12, *7.5*, *11.2*, *11.6*

`undo prevention` *use option* 2.12, *7.5*, *11.2*

The Unexamined Life *example* 13.9 ex 230, *11.3 ex 230*

Unicode *see* characters (letters)

Unicode Character Names / Full Character Names *extension* 5.11, 27.25

`unindexed`, headings in extensions 27.10

units of measure 4.5, 15.1

- arithmetic 15.19
- averages 15.17
- defining 15.8, 15.9, 15.10, 15.14, 15.15
- displaying 15.8, 15.11, 15.15
- for capacity of containers *10.5*
- Metric Units *extension* 15.13, 15.18, 15.20
- multiple notations for a unit 15.9, 15.10, 15.11
- multiplication of units 15.20
- totals 15.17
- understanding 15.16
- with multiple numeric components 15.14, 15.15

`unless` *(a condition) phrase* 11.6

`unless` *(a condition)* `,` *(a phrase) phrase* 11.6

`unlikely / possible / likely, it is unlikely / possible / likely` *(does the player mean) outcome phrase* 17.19

`unlit / lit` *(thing)* *property* 3.15, *3.7*

`unlocked / locked` *(container/door)* *property* 3.13, *9.2*

`unlocks` *relation verb* 3.13

`unmarked for listing / marked for listing` *(thing)* *property* 18.25, *3.1*

`unmentioned / mentioned` *(thing)* *property* 18.24, 18.26, 18.28, *3.1*

`unopenable / openable` *(container)* *property* 3.6, *9.2*

Unpeeled *example* 18.13 ex 344, *9.2 ex 344*

`unpunctuated word number` *(number)* `in` *(text)* *phrase* 20.3

`unsubstituted / substituted` *(text)* *adjective* 20.7

`unsuccessful attempt by` *rulebook* 12.5, 12.12

not used for spontaneous actions 12.6

Unthinkable Alliances *example* 13.9 ex 229, *7.16 ex 229*

`untouchable / touchable` *(thing)* *adjective* 6.13

Up and Up *example* 3.2 ex 6, *6.9 ex 6*

`update backdrop positions` *phrase* 8.8

`upper case --` *(text)* `in upper case` *phrase* 20.4

Uptempo *example* 19.4 ex 394, *4.1 ex 394*

Uptown Girls *example* 8.18 ex 132, *3.9 ex 132*

`"[us]"` *say phrase* 14.4

USE *6.17*

`use` *(options...)* *assertion* 2.12

use options 2.12

active / inactive *adjective* 27.18

catalogue

abbreviated room descriptions 2.12

American dialect 2.12, 2.13, 5.4

authorial modesty 27.6

BRIEF room descriptions 2.12, *6.4*

DICT_WORD_SIZE *2.2*

dynamic memory allocation 20.2

engineering notation 15.2

fast route-finding 6.14

full-length room descriptions 2.12

Inform 6 memory limits 2.14

maximum text length 20.2

maximum things understood at once 4.14

memory economy 2.14

scoring 2.12, 9.2, *11.4*

slow route-finding 6.14

SUPERBRIEF room descriptions 2.12, *6.4*

telemetry recordings 2.13

the serial comma 2.12, 2.13, 5.5, 21.3

unabbreviated object names 3.2

undo prevention 2.12, *7.5*, *11.2*

VERBOSE room descriptions 2.12, *6.4*

defining in Inform 6 27.18

user interface 1.3

facing pages layout 1.3

Go button 1.4

Index panel 1.6

Actions page 12.5, 12.21, 17.1, 17.2, 17.3, 24.3

checking story correctness 24.1

Contents page 2.6, 23.5, 27.10

Kinds page 4.1, 4.6, 4.8, 4.11, 4.12, 27.10

Map page 2.7, 3.4, 3.26, 25.18

Phrases page 2.2, 11.2

Rules page 19.6, 19.16

Scenes page 10.3, 10.9

Release button 2.7, 25.6

Replay button 1.5, 1.7

Results panel 1.6

Skein panel 1.7, 1.8, 1.9, 25.16, *13.1*

Transcript panel 1.8, 1.9

using doors, in route-finding 6.14

usually / seldom, in assertions 4.3

V

value properties 3.10, 4.6

- changing 8.5
- defining 4.8, 4.9, 4.10
- numeric comparatives, adjectives for 6.7, 15.12
- numeric magnitudes, adjectives for 6.6
- numeric superlatives, adjectives for 6.8, 15.12
- used as adjectives 6.3

values

- actions as values 12.20
- basis of rulebooks 19.9
- constant *see* constant values
- descriptions as values 22.2
- displaying 4.9, 5.3
- figures as values 23.10
- in relations 13.13
- kinds of kinds of value 22.1, 22.6
- kinds of value 4.5, 4.9, 4.10
  - comparisons 11.18
  - default values 4.11
  - defining with a table 16.17
  - understanding 17.9, 17.14

phrases as values 22.3

produced by rulebooks 19.13

relations as values 13.14

sounds as values 23.10

tables as values 16.15

truth state values 11.5

understanding 17.6

varying *see* variables: global

Van Helsing *example* 3.25 ex 39, *7.13 ex 39*

variables

changing global variables 8.1

changing temporary variables 11.15

for actions 12.10

for activities 18.6

for rulebooks 19.10

global 4.12

defining in Inform 6 27.22

kind variables 22.1, 22.7

temporary 11.15

displaying 20.7

Variety *example* 14.3 ex 246, *2.1 ex 246*

Variety 2 *example* 14.3 ex 247, *2.1 ex 247*

`various-to-one` *(relation)* *adjective* 13.14

`various-to-various` *(relation)* *adjective* 13.14

`various`, in defining relations 13.5

`"['ve]"` *say phrase* 14.8

vehicles *kind* 3.16, *8.1*

- moving rooms *8.2*
- rules for going by vehicles 7.14

`the verb` *(V)* `means` *(property)* *assertion* 15.12

`the verb` *(V)* `means` *(relation)* *assertion* 13.9

`the verb to be` *(P)* `means` *(relation)* *assertion* 13.10

`VERBOSE room descriptions` *use option* 2.12, *6.4*, *11.5*

Verbosity 1 *example* 3.1 ex 3, *6.4 ex 3*

Verbosity 2 *example* 19.4 ex 395, *6.4 ex 395*

verbs

- adaptive verbs in displayed text 14.2
- as values 14.9
- defining adaptive displayable verbs 14.3
- `meaningful / meaningless` *adjective* 14.9
- `modal / non-modal` *adjective* 14.7, 14.9
- sentence verbs *see* sentence verbs
- understanding verbs in player's command *see* understanding

VERIFY 24.4, *11.5*

Veronica *example* 7.13 ex 100, *6.9 ex 100*

Versailles *example* 6.9 ex 75, *8.5 ex 75*

VERSION 18.37, 23.12, 26.6, 27.5, 27.6, *11.5*

version number (of extension) 27.6

version number (of story) *see* release number

vertical stroke *see* punctuation: vertical stroke, for paragraph break

`very likely / unlikely` *see* likely / possible / unlikely, it is likely / possible / unlikely (does the player mean)

Vesine, Justin de 1.2

victory 9.4, 18.38, 18.39, *11.6*

A View of Green Hills *example* 6.14 ex 80, *3.4 ex 80*

virtual machine 2.14, 23.3

- character set limits 20.4
- extensions for Z-machine / Glulx only 27.7
- Glulx *see* Glulx
- memory limits 2.14, 20.2, 21.10
- multimedia support 23.1, *12.5*
- numeric limits 11.10, 15.2, 15.10, 15.14
- `starting the virtual machine` *activity* 18.40
- Z-machine *see* Z-machine

Virtue *example* 12.3 ex 184, *7.14 ex 184*

`visibility` *rulebook* 12.19

visibility *relation* 6.13

in action processing sequence 12.2

requirement in defining actions 12.7, 12.17, 12.19, 17.1

rules for determining darkness 12.19

`visible / invisible` *(thing)* *adjective* 6.13

Vitrine *example* 8.4 ex 116, *3.6 ex 116*

`volume` *of source text* 2.5

Vorple system *extension* 25.12

Vouvray *example* 4.1 ex 43, *2.2 ex 43*

W

Wainwright Acts *example* 13.12 ex 236, *3.5 ex 236*

`waiting` *action* *6.11*

walkthrough, generated from Skein 25.16

Walls and Noses *example* 18.31 ex 370, *6.17 ex 370*

`was`, for past conditions 9.13

water *see* liquids (implementing)

Waterworld *example* 12.18 ex 216, *3.4 ex 216*

Ways Out *example* 18.23 ex 350, *12.2 ex 350*

`"[we]"` *say phrase* 14.4

`"[wear]"` *say phrase* 14.2

`wearable` *(thing)* *property* 3.20, *9.3*

wearing *relation* 3.20, 8.7, 13.4

Weathering *example* 8.18 ex 131, *3.9 ex 131*

Wesponses *example* 18.15 ex 345, *2.1 ex 345*

Western Art History 305 *example* 17.20 ex 328, *6.15 ex 328*

What Makes You Tick *example* 21.6 ex 427, *9.8 ex 427*

What Not To Wear *example* 13.16 ex 242, *9.3 ex 242*

When? *example* 5.6 ex 63, *3.5 ex 63*

`when` *(condition)*

arbitrary conditions for rules 7.12, 9.5, 19.7

arbitrary conditions for `understand` 17.17

`when play begins` *rulebook* 8.2, 9.1, 19.15, *11.1*

`printing the banner text` *activity* 18.37

`when play ends` *rulebook* 9.4, 19.15, *11.6*

`amusing a victorious player` *activity* 18.39

`printing the player's obituary` *activity* 18.38

`when` *(scene)* `begins / ends` 10.2

Whence? *example* 5.6 ex 64, *3.5 ex 64*

`whether or not` *(a condition)* *phrase* 11.5

`which`, in description clauses 6.9

`which varies`, in defining global variables 4.12

`which -- asking which do you mean` *activity* 18.31

`while` *(a condition)* *phrase* 11.9

`while` *(condition)*

arbitrary conditions for rules 18.4, 19.7

Whither? *example* 17.16 ex 321, *3.5 ex 321*

`who`, in description clauses 6.9

`whole number --` *(real number)* `to the nearest whole number` *phrase* 15.3

Widget Enterprises *example* 15.18 ex 263, *9.4 ex 263*

Wild, Campbell 1.2

Willis, Dannii 1.2

windows, extra story windows *12.5*

winning the story 9.4, 18.38, 18.39, *11.6*

`with nouns reversed` 17.2

`within` *relation verb* *see* containment

`without leading zeroes`, when defining units 15.15

`without printing a room description` 8.7, 8.9

Witnessed 1 *example* 9.5 ex 139, *10.7 ex 139*

Witnessed 2 *example* 11.17 ex 180, *9.11 ex 180*

woman *kind* 3.17, 4.1

`"[won't]"` *say phrase* 14.8

Wonderland *example* 15.8 ex 256, *6.9 ex 256*

Wonka's Revenge *example* 11.10 ex 175, *9.5 ex 175*

`word number` *(number)* `in` *(text)* *phrase* 20.3

The World of Charles S. Roberts *example* 3.26 ex 41, *3.2 ex 41*

`worn` *relation verb* *see* wearing

`worn` *(thing)* *adjective* 3.20, 8.13

`"[would]"` *say phrase* 14.7

Would you...? *example* 4.8 ex 48, *9.1 ex 48*

`"[wouldn't]"` *say phrase* 14.7

`write` *(external file)* `from` *(table name)* *phrase* 23.13

`write` *(text)* `to` *(external file)* *phrase* 23.14

`writing a paragraph about` *activity* 18.24, *3.1*

WXPQ *example* 18.35 ex 380, *6.17 ex 380*

X

Xerxes *example* 18.39 ex 386, *11.6 ex 386*

Xot *example* 18.35 ex 381, *2.3 ex 381*

Xylan *example* 17.1 ex 288, *10.2 ex 288*

XYZZY *example* 17.1 ex 287, *6.13 ex 287*

XYZZY 3.1, 17.1, 20.5, 26.7, 26.10, *6.13*

Y

Y ask Y? *example* 7.16 ex 111, *11.3 ex 111*

`yes` *phrase* 11.16

YES, querying yes/no 11.5

YES, responding to a character *7.7*

Yolk of Gold *example* 6.16 ex 83, *8.4 ex 83*

you, adaptive pronouns in displayed text 14.4

Your Mother Doesn't Work Here *example* 21.11 ex 438, *7.12 ex 438*

`yourself` — *person constant* 3.24, 6.13, 8.9

Z

Z-machine 2.14, 23.3

character set limits 20.2, 20.4

extensions for Z-machine only 27.7

memory limits 2.14, 20.2, 21.10

no support for multimedia 23.1, *12.5*

numeric limits 11.10, 15.2, 15.10, 15.14

real numbers not available 15.2

Zero *example* 9.12 ex 148, *5.5 ex 148*

Zodiac *example* 7.9 ex 96, *7.3 ex 96*

Zorb *example* 7.14 ex 108, *6.9 ex 108*

Zork II *example* 8.19 ex 134, *3.2 ex 134*

Zorn of Zorna *example* 18.18 ex 347, *3.7 ex 347*

Zqlran Era 8 *example* 15.15 ex 260, *4.1 ex 260*